IDENTIFICATION

PRODUCT CODE:	AH-?????-DD
DIAGNOSTIC CODE:	DCKAD
PRODUCT NAME:	DFKADB0 PDP10 KL10 BASIC INSTRUCTION DIAGNOSTIC #4
VERSION:	0.2
DATE RELEASED:	SEPT 1984
MAINTAINED BY:	DIAGNOSTIC ENGINEERING
AUTHOR:	JOHN A. KIRCHOFF

TABLE OF CONTENTS

1.0 ABSTRACT

THIS DECSYSTEM KL10 BASIC INSTRUCTION DIAGNOSTIC IS THE FOURTH IN A SERIES OF DECSYSTEM KL10 PROCESSOR DIAGNOSTICS. THE DIAGNOSTIC TESTS INDEXED ADDRESSING, INDIRECT ADDRESSING, INDEXED-INDIRECT ADDRESSING AND ARITHMETIC FLAGS TESTS. THE DIAGNOSTIC ALSO PERFORMS TESTS OF THE FOLLOWING INSTRUCTIONS: EXCH,MOVEM,JFCL, ADDI,CAI, JRST,JSP, AOBJN, AOBJP,AOS, SOS,JUMPL, JUMPE, JUMPN,JUMPG,AOJ,SOJ, ADDM,HRREM,MOVSM, XORM,ADDB,HLLOS, MOVSS,HRLM,HRRS,HRRZM,SETZB,SETAB,XCT.

THE DIAGNOSTIC TESTS THE BASIC FUNCTIONALITY OF THE R AND MI PROCESSOR.

2.0 REQUIREMENTS

2.1 EQUIPMENT

A KL10 WITH A MINIMUM OF 32K OF MEMORY.

CONSOLE TELETYPE.

2.2 STORAGE

THE PROGRAM RUNS WITHIN 128K OF MEMORY.

2.3 PRELIMINARY PROGRAMS

CONSOLE FUNCTIONS WORKING PROPERLY.
PREVIOUS PROCESSOR DIAGNOSTICS.

3.0 PROGRAM PROCEDURES

3.1 LOADING PROCEDURE

LOAD VIA APPROPRIATE LOADING PROCEDURES.

3.2 STARTING PROCEDURE

STAND-ALONE STARTING ADDRESS IS 40000.

IF THE DIAGNOSTIC FAILS TO START CORRECTLY TRY STARTING AT THE FIRST TEST INSTEAD OF AT THE BEGINNING OF THE CONTROL SEQUENCE. (SEE LISTING).

3.3 OPERATING PROCEDURE

ONCE STARTED THE PROGRAM WILL CYCLE CONTINUALLY UNTIL STOPPED OR AN ERROR OCCURS.

4.0 ERRORS

ERRORS ARE IN THE FORM OF HALT INSTRUCTIONS. THE LISTING SHOULD BE CONSULTED TO DETERMINE THE CAUSE OF THE ERROR. A NO OPERATION (JUMP) INSTRUCTION FOLLOWS EACH HALT. THIS MAY BE USEFUL IN CONSTRUCTING A SCOPE LOOP TO CYCLE ON THE FAILING INSTRUCTION.

5.0 ITERATION COUNTER

THE ITERATION COUNT OF THE PROGRAM CAN BE DETERMINED BY EXAMINING LOCATION "PASCNT".

6.0 CYCLE TIME

THE CYCLE TIME OF THE PROGRAM IS IN THE MILLISECOND RANGE AND IS THEREFORE SUITABLE FOR TAKING MARGINS, VIBRATION TESTS, ETC.

7.0 OPERATIONAL VARIATIONS

A. DIAGNOSTIC MONITOR

THE PROGRAM IS USABLE WITH THE DIAGNOSTIC MONITOR TO PROVIDE RELIABILITY TESTS, ACCEPTANCE TESTS, AND / OR TO PROVIDE A QUICK METHOD OF ISOLATION OF A FAULT TO A PARTICULAR AREA OF THE PROCESSOR. CERTAIN PROCEDURES ARE USED WHEN THE PROGRAM IS USED IN THIS MANNER. THEY ARE:

1. THE DIAGNOSTIC MONITOR TRANSFERS CONTROL TO THE PROGRAM AND STARTS IT AT LOCATION 40002.

2. MONCTL - LOCATION 40064 IS USED AS THE DIAGNOSTIC MONITOR CONTROL FLAG WORD.

B. USER MODE

THE PROGRAM WILL OPERATE IN USER MODE AND AS SUCH PROVIDES ASSURANCE THAT THE PROCESSOR IS PERFORMING ALL FUNCTIONS CORRECTLY. USER MODE STARTING ADDRESS IS 40000.

C. SYSTEM EXERCISER

STARTING ADDRESS IS 40003. NO DATA SWITCHES ARE USED BY THIS PROGRAM.

8.0 MISCELLANEOUS

NONE

9.0 LISTING

HISTORY FILE FOR DCKAD

CODE: MAINDEC-10-DCKAD

TITLE: DECSYSTEM KC10 BASIC INSTRUCTION DIAGNOSTIC (4)

VERSION: 0.2

DATE: August 1982

REASON: Reassembled for changes to SPCCPU.KCM

CODE: MAINDEC-10-DCKAD

TITLE: DECSYSTEM KC10 BASIC INSTRUCTION DIAGNOSTIC (4)

VERSION: 0.1

DATE: March 1982

REASON: ORIGINAL RELEASE OF THIS PROGRAM FOR KC10.

NOTE: THIS PROGRAM WAS ADAPTED FROM 'DSKAD' - AN ANALOGOUS PROGRAM FOR THE DECSYSTEM 2020 KS-10.

```
                                        ;DFKAD
                                        
                    000000              MCNVER==0
                    000002              DECVER==2
                                        
                                                XLIST
                                                LIST
                                                LALL
                                        
                                                NAME    \MCNVER,\DECVER^
                                        
                                        TITLE   DFKAD PDP10 KL10 BASIC INSTRUCTION DIAGNOSTIC (4) 0,2
                                        ^
                                        
                                                ;BYTE, BLOCK TRANSFER, JFFO AND OTHER BASIC MISCELLANEOUS INSTRUCTIONS
                                        
                                        ;COPYRIGHT 1984
                                        ;DIGITAL EQUIPMENT CORPORATION
                                        ;MARLBORO, MASS. 01752
                                        
                                        ;J. Kirchoff
                                        
000137                                          LOC     137
000137  000000  000002                          MCNVER,,DECVER
                                        
                                                NOSYM
```

```
                                        SUBTTL  DIAGNOSTIC PARAMETERS

                                        ;Parameter definitions
                    000040              DEBUG=40
                    000001              EXCASB==1
                    000001              USRASB==1
                    000001              KI10=1
                    000001              KL10=1
                    000001              KL10PO=1
                    000001              PGMEND==1
                    000001              ERDIAG=1
                    000000              MEMMAP==0

                                                ;SPECIAL FEATURE DEFINATIONS

                    030000                      SADR1=BEGIN
                    030000                      SADR2=BEGIN
                    030000                      SADR3=BEGIN
                    030000                      SADR4=BEGIN
            254000  030000                      SADR5=JRST BEGIN
            254000  030000                      SADR6=JRST BEGIN
            254000  030000                      SADR7=JRST BEGIN
            254000  030000                      SADR8=JRST BEGIN
            254000  030000                      SADR9=JRST BEGIN
            254000  030000                      SADR10=JRST BEGIN
            254000  030000                      SADR11=JRST BEGIN


                                                ;SPECIAL FEATURE PARAMETERS

                    000000                      PAREA0=0
                    000000                      PAREA1=0
                    000000                      PAREA2=0
            444653  414400                      PAREA3=SIXBIT/DFKAD/
            546064  000000                      PAREA4=SIXBIT/LPT/
                    000000                      PAREA5=0
                    000000                      PAREA6=0
                    001000                      ITERAT==1000


                                        ;Macros

                                        ; Stop - used for scope loop, if instruction fails, change (jumpa .+1)
                                        ;        To a (jumpa .-x) to cycle on failing instruction

                                        DEFINE  STOP    (A)<
                                                HALT    .+1                     ;Test failed if program halts here
                                                JUMPA   .+1                     ;To loop change this instruction>

                                        ; Sflag - used to clear all flags then to set selected flag

                                        DEFINE  SFLAG   (A)<
                                                MOVSI   1,A
                                                JFCL    17,.+1                  ;Reset all flags
                                                JRST    2,.+1(1)                ;Set a flag>
```

```
                        SUBTTL  *PARAM* CONSOLE DATA SWITCH ASSIGNMENTS, SEPT 18,1979

                        DEFINE  S,<; **********************************************************************>

                        ; **********************************************************************
                        ;*DATA SWITCHES (READ FROM CONSOLE IN EXEC MODE OR TYPED IN IN USER MODE)
                        ;*LEFT HALF SWITCHES ARE PRE-ASSIGNED FOR SUBROUTINE PACKAGE USE
                        ;*AND CONTROL LOOPING, PRINTING (TTY OR OTHER DEVICE) AND MISC. FUNCTIONS
                        ; **********************************************************************

400000          ABORT== 400000          ;ABORT PROGRAM ON PASS COMPLETION
200000          RSTART==200000          ;RESTART TEST, PRINT TOTALS
100000          TOTALS==100000          ;PRINT TOTALS, CONTINUE

040000          NOPNT== 040000          ;INHIBIT ALL PRINT/TYPE OUT (EXCEPT FORCED)
020000          PNTLPT==020000          ;PRINT ALL DATA ON LPT (LOGICAL DEVICE, USER MODE)
010000          DING==  010000          ;RING BELL ON ERROR

004000          LOOPER==004000          ;ENTER EXERCISE/CHECK LOOP ON ERROR
002000          ERSTOP==002000          ;HALT ON TEST ERROR
001000          PALERS==001000          ;PRINT ALL ERRORS

000400          RELIAB==000400          ;RELIABILITY MODE
000200          TXTINH==000200          ;INHIBIT ERROR TEXT
000100          INHPAG==000100          ;INHIBIT PAGING

000040          MODDVC==000040          ;MODIFY DEVICE CODE
000020          INHCSH==000020          ;INHIBIT CACHE
000010          OPRSEL==000010          ;OPERATOR SELECTION

000004          CHAIN== 000004          ;CHAIN CONTROL SWITCH

000002          KAHZ50==000002          ;KA10 50 HERTZ POWER

                                        ;SWITCH 17 RESERVED !!!
```

```
                        SUBTTL  *PARAM* PROGRAM/SUBROUTINE PARAMETERS, SEPT 18,1979

                        ; ************************************************************************
                        ;*SPECIAL SUBPROGRAM LINKAGES
                        ; ************************************************************************

027772                  FSELNK= 27772   ;FILE SELECT LINK
027773                  FRDLNK= 27773   ;FILE READ LINK
027774                  LDLNK=  27774   ;LOAD LINKAGE ADDRESS
027775                  DDTLNK= 27775   ;DDT LINKAGE ADDRESS
027776                  MODLNK= 27776   ;OPERATIONAL MODE CHECK LINKAGE ADDRESS
027777                  SUBLNK= 27777   ;SUBROUTINE LINKAGE ADDRESS

                        ; ************************************************************************
                        ;*SPECIAL SUBROUTINE FATAL HALTS
                        ;*USED TO REPORT ERRORS THAT CAUSE THE SUBROUTINES TO BE UNUSABLE
                        ; ************************************************************************

                        ;ADDRESS   TAG  REASON
                        ;--------------------

                        ; 1010  NOEXEC  ;PROGRAM NOT CODED FOR EXEC MODE OPERATION
                        ; 1011  PLERR   ;FATAL PUSH LIST POINTER ERROR
                        ; 1012  PLERR1  ;INITIAL PUSH LIST POINTER ERROR
                        ; 1013  MUOERR  ;MUUO WITH LUUO HANDLER WIPED OUT
                        ; 1014  DTEBER  ;DTE20 INTERRUPT WITHOUT DOORBELL
                        ; 1015  DTECER  ;DTE20 CLOCK INTERRUPT WITHOUT FLAG SET
                        ; 1016  CPIERR  ;CPU INITIALIZATION ERROR
                        ; 1017  EOPERR  ;END OF PROGRAM ERROR
                        ; 1020  LUOERR  ;INTERRUPT WITH LUUO HANDLER WIPED OUT

                        ; ************************************************************************
```

```
                        ; ************************************************************************
                        ;OPERATOR DEFINITIONS (NON-UUO'S)
                        ; ************************************************************************

260740  000000          OPDEF   GO      [PUSHJ  P,]     ;SUBROUTINE CALL
263740  000000          OPDEF   RTN     [POPJ   P,]     ;SUBROUTINE RETURN
261740  000000          OPDEF   PUT     [PUSH   P,]     ;PUT DATA ON PUSH LIST
262740  000000          OPDEF   GET     [POP    P,]     ;GET DATA FROM PUSH LIST
254000  000000          OPDEF   PJRST   [JRST   ]       ;JRST TO ROUTINE THAT RTN'S
254200  000000          OPDEF   HALT    [JRST   4,]     ;DEFINITION FOR DDT
254100  000000          OPDEF   JRSTF   [JRST   2,]     ;DEFINITION FOR DDT
254500  000000          OPDEF   JEN     [JRST   12,]    ;DEFINITION FOR DDT

                        ; ************************************************************************
                        ;SUBROUTINE INITIALIZATION CALL
                        ; ************************************************************************

265000  030011          OPDEF   PGMINT  [JSP    0,SBINIT]       ;SUBROUTINE INITIALIZATION

                        ; ************************************************************************
                        ;HALTING UUO'S (A MORE GRACEFUL HALT THAN SIMPLY USING THE HALT INSTRUCTION).
                        ; ************************************************************************

037640  000004          OPDEF   FATAL   [37B8!15B12!4]  ;FATAL PROGRAMMING HALT
037600  000004          OPDEF   ERRHLT  [37B8!14B12!4]  ;PROGRAM ERROR HALT

                        ; ************************************************************************
                        ;TERMINAL INPUT UUO'S
                        ;ALWAYS COME FROM THE CONSOLE TERMINAL IN EXEC MODE OR THE
                        ;CONTROLLING TERMINAL (REAL TERMINAL OR PTY) IN USER MODE.
                        ; ************************************************************************

037000  000003          OPDEF   TTICHR  [37B8!0B12!3]   ;TTY, INPUT ANY CHARACTER
037040  000003          OPDEF   TTIYES  [37B8!1B12!3]   ;TTY, NORMAL RETURN Y
037100  000003          OPDEF   TTINO   [37B8!2B12!3]   ;TTY, NORMAL RETURN N
037140  000003          OPDEF   TTIOCT  [37B8!3B12!3]   ;TTY, INPUT OCTAL WORD
037200  000003          OPDEF   TTIDEC  [37B8!4B12!3]   ;TTY, INPUT DECIMAL WORD
037240  000003          OPDEF   TTICNV  [37B8!5B12!3]   ;TTY, INPUT CONVERTABLE WORD
037300  000003          OPDEF   TTLOOK  [37B8!6B12!3]   ;TTY, KEYBOARD CHECK
037340  000003          OPDEF   TTALTM  [37B8!7B12!3]   ;TTY, ALT-MODE CHECK
037400  000003          OPDEF   TTSIXB  [37B8!10B12!3]  ;TTY, INPUT SIXBIT WORD
037440  000003          OPDEF   TTYINP  [37B8!11B12!3]  ;TTY, IMAGE MODE INPUT
037500  000003          OPDEF   TTICLR  [37B8!12B12!3]  ;TTY, CLEAR INPUT
```

```
                                ;TERMINAL OUTPUT UUO'S.

            037000  000000      OPDEF   PNTA    [37B8!0B12!0]   ;PRINT ASCII WORD
            037000  000001      OPDEF   PNTAF   [37B8!0B12!1]   ;PRINT ASCII WORD FORCED
            037740  000000      OPDEF   PNTAL   [37B8!17B12!0]  ;PRINT ASCIZ LINE
            037740  000001      OPDEF   PNTALF  [37B8!17B12!1]  ;PRINT ASCIZ LINE FORCED
            037600  000003      OPDEF   PSIXL   [37B8!14B12!3]  ;PRINT SIXBIT'Z LINE
            037640  000003      OPDEF   PSIXLF  [37B8!15B12!3]  ;PRINT SIXBIT'Z LINE FORCED
            037000  000000      OPDEF   PNTMSG  [37B8!0B12!0]   ;PRINT MESSAGE IMMEDIATE
            037040  000000      OPDEF   PNTMSF  [37B8!1B12!0]   ;PRINT MESSAGE IMMEDIATE FORCED
            037100  000000      OPDEF   PSIXM   [37B8!2B12!0]   ;PRINT SIXBIT'Z MSG IMMEDIATE
            037200  000000      OPDEF   PSIXMF  [37B8!4B12!0]   ;PRINT SIXBIT'Z MSG IMM FORCED
            037000  000000      OPDEF   PNTCI   [37B8!0B12!0]   ;PRINT CHARACTER IMMEDIATE
            037040  000000      OPDEF   PNTCIF  [37B8!1B12!0]   ;PRINT CHARACTER IMMEDIATE FORCED
            037500  000000      OPDEF   PNTCHR  [37B8!12B12!0]  ;PRINT CHARACTER
            037500  000001      OPDEF   PNTCHF  [37B8!12B12!1]  ;PRINT CHARACTER FORCED
            037040  000000      OPDEF   PNT1    [37B8!1B12!0]   ;PRINT ONE OCTAL DIGIT
            037040  000001      OPDEF   PNT1F   [37B8!1B12!1]   ;PRINT 1 OCTAL DIGIT FORCED
            037100  000000      OPDEF   PNT2    [37B8!2B12!0]   ;PRINT TWO OCTAL DIGITS
            037100  000001      OPDEF   PNT2F   [37B8!2B12!1]   ;PRINT 2 OCTAL DIGITS FORCED
            037140  000000      OPDEF   PNT3    [37B8!3B12!0]   ;PRINT THREE OCTAL DIGITS
            037140  000001      OPDEF   PNT3F   [37B8!3B12!1]   ;PRINT THREE OCTAL DIGITS FORCED
            037200  000000      OPDEF   PNT4    [37B8!4B12!0]   ;PRINT FOUR OCTAL DIGITS
            037200  000001      OPDEF   PNT4F   [37B8!4B12!1]   ;PRINT FOUR OCTAL DIGITS FORCED
            037240  000000      OPDEF   PNT5    [37B8!5B12!0]   ;PRINT FIVE OCTAL DIGITS
            037240  000001      OPDEF   PNT5F   [37B8!5B12!1]   ;PRINT FIVE OCTAL DIGITS FORCED
            037300  000000      OPDEF   PNT6    [37B8!6B12!0]   ;PRINT SIX OCTAL DIGITS
            037300  000001      OPDEF   PNT6F   [37B8!6B12!1]   ;PRINT SIX OCTAL DIGITS FORCED
            037340  000000      OPDEF   PNT7    [37B8!7B12!0]   ;PRINT 7 OCTAL DIGITS
            037340  000001      OPDEF   PNT7F   [37B8!7B12!1]   ;PRINT 7 OCTAL DIGITS FORCED
            037440  000000      OPDEF   PNT11   [37B8!11B12!0]  ;PRINT 11 OCTAL DIGITS
            037440  000001      OPDEF   PNT11F  [37B8!11B12!1]  ;PRINT 11 OCTAL DIGITS FORCED.
            037400  000000      OPDEF   PNTADR  [37B8!10B12!0]  ;PRINT PHYSICAL ADDRESS
            037400  000001      OPDEF   PNTADF  [37B8!10B12!1]  ;PRINT PHYSICAL ADDRESS FORCED
            037600  000000      OPDEF   PNTOCT  [37B8!14B12!0]  ;PRINT FULL WORD OCTAL
            037600  000001      OPDEF   PNTOTF  [37B8!14B12!1]  ;PRINT FULL WORD OCTAL FORCED
            037540  000000      OPDEF   PNTHW   [37B8!13B12!0]  ;PRINT OCTAL HALF WORDS, 6 SP 6
            037540  000001      OPDEF   PNTHWF  [37B8!13B12!1]  ;PRINT OCTAL HALF WORDS, 6 SP 6 FORCED
            037700  000003      OPDEF   PNTOCS  [37B8!16B12!3]  ;PRINT OCTAL, SUPPRESS LEADING 0'S
            037740  000003      OPDEF   PNTOCF  [37B8!17B12!3]  ;PRINT OCTAL, SUPPRESS LEADING 0'S FORCED
            037640  000000      OPDEF   PNTDEC  [37B8!15B12!0]  ;PRINT DECIMAL, SUPRESS LEADING 0'S
            037640  000001      OPDEF   PNTDCF  [37B8!15B12!1]  ;PRINT DECIMAL, SUPRESS LEADING 0'S FORCED
            037700  000000      OPDEF   PNTDS   [37B8!16B12!0]  ;PRINT DECIMAL, SPACES FOR LD 0'S
            037700  000001      OPDEF   PNTDSF  [37B8!16B12!1]  ;PRINT DECIMAL, SPACES FOR LD 0'S FORCED
```

```
                037200  000002          OPDEF   PNTNM   [37B8!4B12!2]   ;PRINT PROGRAM NAME
                037000  000002          OPDEF   PNTSIX  [37B8!0B12!2]   ;PRINT SIXBIT WORD
                037040  000002          OPDEF   PNTSXF  [37B8!1B12!2]   ;PRINT SIXBIT WORD FORCED
                037240  000002          OPDEF   DROPDV  [37B8!5B12!2]   ;CLOSE LOGICAL FILE, USER MODE
                037100  000002          OPDEF   PNTCW   [37B8!2B12!2]   ;PRINT DF10 CONTROL WORD
                037140  000002          OPDEF   PNTCWF  [37B8!3B12!2]   ;PRINT DF10 CONTROL WORD FORCED
                037000  030242          OPDEF   PCRL    [37B8!0B12!CRLF] ;PRINT CARRIAGE RETURN/LINE FEED
                037040  030242          OPDEF   PCRLF   [37B8!1B12!CRLF] ;PRINT CARRIAGE RETURN/LINE FEED FORCED
                037000  000040          OPDEF   PSP     [37B8!0B12!40]  ;PRINT SPACE
                037040  000040          OPDEF   PSPF    [37B8!1B12!40]  ;PRINT SPACE FORCED
                037000  030243          OPDEF   PCRL2   [37B8!0B12!CRLF2] ;PRINT CARRIAGE RETURN/LINE FEED (TWICE)
                037040  030243          OPDEF   PCRL2F  [37B8!1B12!CRLF2] ;PRINT CARRIAGE RETURN/LINE FEED (TWICE) FORCED
                037040  000007          OPDEF   PBELL   [37B8!1B12!7]   ;PRINT TTY BELL

                037040  000026          OPDEF   PFORCE  [37B8!1B12!26]  ;PRINT FORCE, CONTROL O OVERRIDE

                                        DEFINE  PMSG    (ARG),<
                                                PSIXM   [SIXBIT\ARG'_\]>

                                        DEFINE  PMSGF   (ARG),<
                                                PSIXMF  [SIXBIT\ARG'_\]>

                                        ;SIXBTZ -- MACRO TO GENERATE SIXBIT DATA FOR PRINTING
                                        ;       CONSERVES CORE OVER ASCIZ

                                        DEFINE  SIXBTZ  (ARG),< [SIXBIT\ARG'_\]>

                                        ;CONSOLE SWITCH INPUT UUO.
                                        ;READS CONSOLE SWITCHES IF IN EXEC MODE OR ASKS FOR THEM IF
                                        ; USER MODE.

                037400  000002          OPDEF   SWITCH  [37B8!10B12!2]  ;INPUT CONSOLE SWITCHES

                                        ;CLOCK INITIALIZATION UUO - TO SET DESIRED CLOCK OPERATION
                                        ;EITHER IGNORE CLOCK, ONLY LET IT TICK OR CAUSE INTERRUPT TO OCCUR.

                037540  000004          OPDEF   CLOKOP  [37B8!13B12!4]  ;CLOCK OPERATION UUO - DP-11 CLOCK
                037200  000004          OPDEF   MTROP   [37B8!4B12!4]   ;CLOCK OPERATION UUO - KA20 METER

                                        ;KL10 ONLY CACHE OPERATION UUO'S

                037040  000004          OPDEF   CINVAL  [37B8!1B12!4]   ;CACHE INVALIDATE
                037100  000004          OPDEF   CFLUSH  [37B8!2B12!4]   ;CACHE FLUSH
                037140  000004          OPDEF   CWRTBI  [37B8!3B12!4]   ;CACHE WRITE-BACK & INVALIDATE
```

```
                                ;END OF PASS/PROGRAM UUOS

                                ;PERFORMS THE END OF PASS FUNCTIONS. INCREMENT PASS COUNT,
                                ;DECREMENT ITERATION COUNT, CHECK IF FINISHED WITH THIS PROGRAM ETC.

037500  000004                  OPDEF   ENDUUO  [37B8!12B12!4]  ;UUO TO DISPLAY LIGHTS
037700  000004                  OPDEF   EOPUUO  [37B8!16B12!4]  ;END OF PROGRAM UUO

                                ;MEMORY MANAGEMENT UUO'S
                                ;UUO'S TO PERFORM VARIOUS MEMORY FUNCTIONS. MAPPING, ZEROING, PAGING,
                                ;ADDRESS CONVERSION, ETC...

037000  000004                  OPDEF   MAPMEM  [37B8!0B12!4]   ;MAP MEMORY
037500  000002                  OPDEF   MEMZRO  [37B8!12B12!2]  ;ZERO MEMORY
037440  000002                  OPDEF   MEMSEG  [37B8!11B12!2]  ;SETUP MEMORY SEGMENT
037540  000002                  OPDEF   MAPADR  [37B8!13B12!2]  ;VIRTUAL TO PHYSICAL ADR CONVERT
037640  000002                  OPDEF   MAPCNK  [37B8!15B12!2]  ;MAP MEMORY CHUNK
037600  000002                  OPDEF   MAPSET  [37B8!14B12!2]  ;SET KI10 EXEC PAGE MAP
037740  000002                  OPDEF   MAPPNT  [37B8!17B12!2]  ;PRINT MEMORY MAP

                                ;DEVICE CODE MODIFICATION UUO
                                ;ALLOWS THE MODIFICATION OF IOT'S TO ONE DEVICE TO BE CHANGED TO
                                ;IOT'S TO A DIFFERENT DEVICE CODE.

037340  000002                  OPDEF   MODPCU  [37B8!7B12!2]   ;MODIFY PERHIPERAL CODE, USER
037300  000002                  OPDEF   MODPCP  [37B8!6B12!2]   ;MODIFY PERHIPERAL CODE, PROGRAM

        030000                          IFNDEF  MODDVL,<MODDVL==BEGIN>
        030000                          IFNDEF  MODDVU,<MODDVU==BEGIN>

                                ;"DIAMON" FILE SELECTION AND READ UUOS

037240  000004                  OPDEF   FSELECT [37B8!5B12!4]   ;FILE SELECTION
037300  000004                  OPDEF   FREAD   [37B8!6B12!4]   ;FILE READ - ASCII DATA
037340  000004                  OPDEF   FRD36   [37B8!7B12!4]   ;FILE READ - 36 BIT DATA
037400  000004                  OPDEF   FRD8    [37B8!10B12!4]  ;FILE READ - 8 BIT DATA

                                ;KI10 ONLY UUO FOR PRINTING MARGIN VALUES

037700  000002                  OPDEF   PNTMGN  [37B8!16B12!2]  ;PRINT MARGIN VALUE

                                XLIST
                                IFNDEF  KLOLD,<LIST
```

```
                        SUBTTL  ERROR HANDLING UUO DEFINITIONS, SEPT 18,1979

                        ; ************************************************************************
                        ;ERROR HANDLER PARAMETERS
                        ; ************************************************************************

036000  000000          OPDEF   ERUUO   [36B8]          ;ERROR CALL UUO
035000  000000          OPDEF   ERLOOP  [35B8]          ;ERROR LOOP, CHECKS PC,REPT,REPT1,ERROR
035040  000000          OPDEF   ERLP1   [35B8!1B12]     ;ERROR LOOP IF PC'S MATCH
035100  000000          OPDEF   ERLP2   [35B8!2B12]     ;ERROR LOOP IF ANY ERROR
034000  000000          OPDEF   REPTUO  [34B8]          ;REPEAT LOOP UUO

                        ;THE ERROR HANDLER MACROS

                        ;A MACRO TO REPORT AN ERROR AND LOOP

                                DEFINE  ERROR   (ADR,FORMAT,CORECT,ACTUAL,F,D,ERR)<
                                                                        SALL
                                ERUUO   FORMAT,[T,,[SIXBIT\F'_\]
                                           CORECT,,ACTUAL
                                           [SIXBIT\D'_\],,ERR]
                                                                        XALL
                                ERLOOP  ADR             ;IF ERROR, LOOP TO ADR
                        >

                        ;A MACRO TO REPORT AN ERROR AND NOT LOOP

                                DEFINE  ERROR1  (FORMAT,CORECT,ACTUAL,F,D,ERR)<
                                                                        SALL
                                ERUUO   FORMAT,[T,,[SIXBIT\F'_\]
                                           CORECT,,ACTUAL
                                           [SIXBIT\D'_\],,ERR]
                                                                        XALL>

                        >;END OF KLOLD CONDITIONAL

                                XLIST
                                LIST
```

```
                                SUBTTL  *FIXED* FIXED CONTROL AND DISPATCH STORAGE, SEPT 18,1979

030000                                  LOC     30000

                                ; ************************************************************************
                                ;PROGRAM STARTING ADDRESSES
                                ;THESE ADDRESSES CALL VARIOUS SPECIAL START ROUTINES AND OR OPTIONS
                                ;NORMAL START ADDRESS IS 30000 ALL OTHERS ARE SPECIAL. INVOKED BECAUSE
                                ;OF END OF PASS, POWER FAILURE, DDT START, RE-ENTERING(TYPICALLY USER
                                ;MODE), OR ANY NUMBER OF SPECIAL FEATURE TESTS.
                                ; ************************************************************************

030000  254 00 1 00 027776    BEGIN:  JRST    @MODLNK         ;STAND-ALONE START
030001  254 00 0 00 030674    $START: JRST    START           ;MODE CHECK STARTING ADDRESS

030002  254 00 1 00 027774    DIAGMN: JRST    @LDLNK          ;DIAGNOSTIC MONITOR START

030003  254 00 1 00 027774    SYSEXR: JRST    @LDLNK          ;SYSTEM EXERCISER START

030004  254 00 0 00 030000    SFSTRT: JRST    SADR1           ;SPECIAL FEATURE START

030005  254 00 0 00 030000    PFSTRT: JRST    SADR2           ;POWER FAIL RESTART

030006  254 00 0 00 030000    REENTR: JRST    SADR3           ;RE-ENTER START(USUALLY USER MODE ONLY)

030007                        SRTDDT:                         ;COMMONLY MISTAKEN NAME FOR "DDTSRT"
030007  254 00 1 00 027775    DDTSRT: JRST    @DDTLNK         ;DDT START

030010  254 00 0 00 030715    BEGIN1: JRST    STARTA          ;LOOP START(END OF PASS COMES HERE)
030011  254 00 1 00 027777    SBINIT: JRST    @SUBLNK         ;PMGINT LINKAGE
030012  000000  000000        RETURN: 0                       ;RETURN ADDRESS STORAGE

030013  254000  030000        START1: SADR7                   ;OPTIONAL STARTING ADR/INSTRUCTIONS
030014  254000  030000        START2: SADR8                   ;  "
030015  254000  030000        START3: SADR9                   ;  "
030016  254000  030000        START4: SADR10                  ;  "
030017  254000  030000        START5: SADR11                  ;  "
```

```
                                ; ************************************************************************
                                ;PROGRAM FIXED PARAMETER AREA
                                ; ************************************************************************

030020  444653  414400          PNTNAM: PAREA3          ;SIXBIT PROGRAM NAME
030021  546064  000000          PNTEXT: PAREA4          ;SIXBIT PROGRAM EXTENSION
030022  000000  000000          RANDBS: PAREA1          ;RANDOM BASE NUMBER
030023  000000  000000          SWTEXR: PAREA2          ;SYSTEM EXERCISER SWITCHES
030024  000000  001000          ITRCNT: ITERAT          ;PROGRAM ITERATIONS
030025  000000  030701          $PNAME: PGMNAM          ;POINTER TO PROGRAMS NAME
030026  000000  000002          $PVER:  MCNVER,,DECVER  ;MCN & DEC VERSION LEVEL
030027  000000  030000          $MODVL: MODDVL          ;DEVICE CODE CHANGE LOWER LIMIT
030030  000000  030000          $MODVU: MODDVU          ;DEVICE CODE CHANGE UPPER LIMIT
030031  777777  777777          $EMODE: IFNDEF EXCASB,<0> IFDEF EXCASB,<-1>     ;EXEC ALLOWED
030032  777777  777777          $UMODE: IFNDEF USRASB,<0> IFDEF USRASB,<-1>     ;USER ALLOWED
030033  000000  000000          $DSKUP: IFNDEF DSKUPD,<0> IFDEF DSKUPD,<-1>     ;DISK UPDATE MODE
030034  777777  777777          $MMAP:  IFNDEF MEMMAP,<0> IFDEF MEMMAP,<-1>     ;ALLOW MEMORY RTNS
030035  000000  000000          PAREA7: PAREA5          ;OPTIONAL PARAMETER
030036  000000  000000          PAREA8: PAREA6          ;OPTIONAL PARAMETER

                                ; ************************************************************************
                                ;PROGRAM VARIABLE PARAMETER AREA
                                ; ************************************************************************

030037  000000  000000          USER:   0               ; 0 = EXEC, -1 = USER MODE FLAG
030040  000000  000000          KAIFLG: 0               ;PROCESSOR TYPE, 0 = KA10, -1 = KI10
030041  000000  000000          KLFLG:  0               ;PROCESSOR TYPE, 0 = KA/KI, -1 = KL10
030042  777777  777777          MONFLG: -1              ;DIAG MONITOR SPECIAL USER FLAG
030043  000000  000000          MONCTL: 0               ;DIAG MON/SYS EXR FLAG
030044  000000  000000          MONTEN: 0               ;-1= LOADED BY 10
030045  000000  000000          CLOCKF: 0               ;CLOCK TICKED FLAG
030046  000000  000000          CONSW:  0               ;CONSOLE SWITCH SETTINGS
030047  000000  000000          PASCNT: 0               ;PROGRAM PASS COUNT
030050  000000  000000          RUNFLG: 0               ;PROGRAM RUN FLAG
030051  000000  000000          TESTPC: 0               ;SUBTEST PC
030052  000000  000000          ERRPC:  0               ;ERROR PC
030053  000000  000000          ERRTLS: 0               ;ERROR TOTALS
030054  000000  000000          TICKS:  0               ;PROGRAM RUNNING TIME
030055  000000  000000          MARGIN: 0               ;KI10 MARGIN WORD VALUE
030056  000000  000000          $ONETM: 0               ;SUBROUTINE INITIALIZATION FLAG
```

```
                                ; ************************************************************************
                                ;SPECIAL PROGRAM DISPATCH ADDRESSES
                                ; ************************************************************************

030057  037 12 0 00 000004      BEGEND: ENDUUO                  ;END OF PASS
030060  254 00 0 00 030010      $BEND1: JRST    BEGIN1          ;KEEP RUNNING PROGRAM
030061  037 16 0 00 000004      $BEND2: EOPUUO                  ;END OF PROGRAM - NO RETURN
030062  254000  030000          CNTLC:  SADR5                   ;CONTROL C XFER ADDRESS
030063  254000  030000          ALTMGO: SADR6                   ;ALTMODE XFER ADDRESS
030064                          CPOPJ1:                         ;SKIP RETURN
030064  350 00 0 17 000000      UUOSKP: AOS     (P)             ;SKIP RETURN FROM UUO
030065                          CPOPJ:                          ;NON-SKIP REGULAR RETURN
030065  263 17 0 00 000000      UUOEXT: RTN                     ;UUO RETURN
030066  255 00 0 00 000000      UUORTN: JFCL                    ;ADDITIONAL USERS UUO ROUTINE
030067  255 00 0 00 000000      $UORTX: JFCL                    ;ADDITIONAL UUO LINKAGE
030070  255 00 0 00 000000      $UUOER: JFCL                    ;INITED AS (JRST $UOERX)
030071  255 00 0 00 000000      $ITRHL: JFCL                    ;ADDITIONAL INTERRUPT LINKAGE
030072  255 00 0 00 000000      $ITRX1: JFCL                    ; ''
030073  255 00 0 00 000000      $USRHL: JFCL                    ; ''
030074  255 00 0 00 000000      $RSRTX: JFCL                    ;ADDITIONAL POWER FAIL LINKAGE
030075  255 00 0 00 000000      $RSRTY: JFCL                    ; ''
030076  255 00 0 00 000000      RESRT1: JFCL                    ; INITED AS (JRST RESRTX)
030077  255 00 0 00 000000      RESRT2: JFCL                    ; ''
030100  255 00 0 00 000000      $PARER: JFCL                    ;ADDITIONAL PARITY ERROR LINKAGE
030101  255 00 0 00 000000      ERMORE: JFCL                    ;ADDITIONAL ERROR HANDLER LINKAGE
030102  254 04 0 00 030102              HALT    .               ;IMPROPER TRANSFER HALT

030103  000000  000000          $PSHER: 0                       ;INITED AS (JRST PSHERR)
030104  000000  000000          ITRCH1: 0                       ;PC & FLAGS OF CURRENT INTERRUPT
030105  000000  000000                  0                       ;INITED AS (JRST $ITRC1)

                                ; ************************************************************************
                                ;PROCESSOR CONTROL STORAGE
                                ; ************************************************************************

030106  000000  000000          $ACC0:  0               ;INTERRUPT SAVED AC0
030107  000000  000000          $SVPI:  0               ;INTERRUPT SAVED PI
030110  000000  000000          $SVAPR: 0               ;INTERRUPT SAVED APR
030111  000000  000000          $SVPAG: 0               ;INTERRUPT SAVED PAG (DATAI)
030112  000000  000000          $SPAG1: 0               ;INTERRUPT SAVED PAG (CONI)

030113  000000  000000          $SVUUO: 0               ;CURRENT USERS UUO
030114  000000  000000          $SVUPC: 0               ;PC OF CURRENT USERS UUO

030115  000000  000000          REPTU:  0               ;REPEAT UUO ITERATIONS
030116  000000  000000          SCOPE:  0               ;ERROR HANDLER SCOPE LOOP FLAG
030117  000000  000000          %CORFLG:0               ; '' CORRECT FLAG
030120  000000  000000          %COREC: 0               ; '' CORRECT DATA
030121  000000  000000          %ACTFL: 0               ; '' ACTUAL FLAG
030122  000000  000000          %ACTUL: 0               ; '' ACTUAL DATA
030123  000000  000000          %DISCR: 0               ; '' DISCREPENCY DATA
```

```
                                ; ************************************************************************
                                ;UUO DISPATCH TABLE
                                ; ************************************************************************
                                        XLIST
                                        LIST
030124  030070  030070          UUODIS: LUU01,,$UUOER
030125  030070  030070                  LUU03,,LUU02
030126  030070  030070                  LUU05,,LUU04
030127  030070  030070                  LUU07,,LUU06
030130  030070  030070                  LUU011,,LUU010
030131  030070  030070                  LUU013,,LUU012
030132  030070  030070                  LUU015,,LUU014
030133  030070  030070                  LUU017,,LUU016
030134  030070  030070                  LUU021,,LUU020
030135  030070  030070                  LUU023,,LUU022
030136  030070  030070                  LUU025,,LUU024
030137  030070  030070                  LUU027,,LUU026
030140  030070  030070                  LUU031,,LUU030
030141  030070  030070                  LUU033,,LUU032

                                ; ************************************************************************
                                ;MEMORY MANAGMENT STORAGE
                                ; ************************************************************************

030142  000000  000000          DF22F:  0               ;DF10 CONTROL FLAG, 0 = 18, -1 = 22 BIT
030143  000000  000000          MAPNEW: 0               ;MEMORY MAPPING CONTROL FLAG, -1 = 4096K MAPPING
030144  000000  000000          MEMTOT: 0               ;TOTAL MEMORY SIZE IN K (1024.)
030145  000000  000000          MEMLOW: 0               ;LOWEST USABLE MEMORY
030146                          MEMSIZ: BLOCK ^D41      ;MEMORY SEGMENT POINTER TABLE

                                ; ************************************************************************
                                ;PRINT CONTROL STORAGE
                                ; ************************************************************************

030217  000000  000000          PNTFLG: 0               ;PRINT FLAG, -1 WHILE IN PRINT ROUTINE
030220  000000  000000          PNTENB: 0               ;PRINT ENABLE
030221  000000  000000          PDISF:  0               ;PRINT DISABLED FLAG
030222  000000  000000          PNTINH: 0               ;INHIBIT PRINT INPUT CHECKS
030223  000000  000000          PNTSPC: 0               ;PRINT SPACE CONTROL
030224  000000  000000          OPTIME: 0               ;TYPE-IN WAIT TIME
030225  000000  000000          $TWCNT: 0               ;TIME WAITED
030226  000000  000000          $DVOFF: 0               ;LOGICAL DEVICE INITED FLAG
030227  000000  000000          TTYFIL: 0               ;TTY EXEC FILLERS FLAG
030230  000000  000000          TTYSPD: 0               ;TTY EXEC BAUD RATE
030231  000000  000000          $TTCHR: 0               ;ACTUAL TYPED IN CHAR
030232  000000  000000          $CHRIN: 0               ;UPPER CASED & PARITY STRIPPED CHAR
030233  000000  000000          $TYPNB: 0               ;TYPED IN NUMBER
030234  000000  000000          $CRLF:  0               ;FREE CR/LF FLAG
030235  000000  000000          $TABF:  0               ;TAB CONVERSION FLAG
030236  000000  000000          $FFF:   0               ;FORM FEED CONVERSION FLAG
030237  000000  000000          $VTF:   0               ;VERTICAL TAB CONVERSION FLAG
030240  000000  000000          USRLFF: 0               ;USER LF FILLERS
030241  000000  000000          USRCRF: 0               ;USER CR FILLERS
```

```
                                ; ************************************************************************
                                ;THE FOLLOWING MISCELLANEOUS PRINT CHARACTERS ARE INCLUDED
                                ;TO FACILITATE PRINTING AND ARE CALLED AS FOLLOWS:
                                ;       MOVEI   NAME
                                ;       PNTA            ;OR PNTAF
                                ; ************************************************************************

030242                          CRLF:   ASCII/
030242  015 012 000 000 000     /
030243                          CRLF2:  ASCII/

030243  015 012 015 012 000     /
030244  054 000 000 000 000     COMMA:  ASCII/,/
030245  056 000 000 000 000     PERIOD: ASCII/./
030246  040 000 000 000 000     SPACE:  ASCII/ /
030247  011 000 000 000 000     TAB:    ASCII/  /
030250                          MINUS:
030250  055 000 000 000 000     HYPEN:  ASCII/-/
030251  053 000 000 000 000     PLUS:   ASCII/+/
030252  052 000 000 000 000     AST:    ASCII/*/
030253  100 000 000 000 000     ATSIN:  ASCII/@/
030254  050 000 000 000 000     LFP:    ASCII/(/
030255  051 000 000 000 000     RTP:    ASCII/)/
030256  007 0000000000          BELL:   BYTE (7) 007
030257  077 000 000 000 000     QUEST:  ASCII/?/
030260  057 000 000 000 000     SLASH:  ASCII!/!
030261  044 000 000 000 000     DOLLAR: ASCII/$/
030262  000000  000012          RADIX:  ^D10                    ;DECIMAL PRINT RADIX
030263  000000  000040          RADLSP: 40                      ;DECIMAL PRINT LEADING CHAR
030264  000000  000012          RADLSC: ^D10                    ;DECIMAL PRINT LEADING CHAR COUNT

                                ; ************************************************************************
                                ;USER MODE OUTPUT FILE INFORMATION
                                ; ************************************************************************

030265                          $OBUF:  BLOCK   3               ;LOGICAL FILE OUTPUT BUFFER HEADER
030270  60 62 51 56 64 00       $OUTNM: SIXBIT  /PRINT/         ;FILE NAME
030271  60 56 64 00 00 00       $OUTEX: SIXBIT  /PNT/           ;FILE NAME EXTENSION
030272                                  BLOCK   2

                                ; ************************************************************************
                                ;DISK UPDATE MODE FILE INFORMATION
                                ; ************************************************************************

030274                          $IBUF:  BLOCK   3
030277  60 62 51 56 64 00       $INNM:  SIXBIT  /PRINT/
030300  60 56 64 00 00 00       $INEXT: SIXBIT  /PNT/
030301                                  BLOCK   2
```

```
                                ; ***************************************************************************
                                ;PUSHDOWN LIST CONTROL INFORMATION
                                ; ***************************************************************************

030303  777577  030303          PLIST:  PLIST-PLISTE,,PLIST
030304                          PLISTS: BLOCK   200
030504  000000  000000          PLISTE: 0               ;END OF PUSHDOWN LIST

                                ; ***************************************************************************
                                ;POWER LINE CLOCK FREQUENCY FLAG
                                ; ***************************************************************************

030505  000000  000000          CYCL60: 0       ;0 = 60, -1 = 50 CYCLE

                                ; ***************************************************************************
                                ;KL10 CACHE CONTROL FLAGS
                                ; ***************************************************************************

030506  000000  000000          CSHFLG: 0       ;ALLOW CACHE IF 0
030507  000000  000000          CSHMEM: 0       ;CACHE MEMORY SEGMENTS IF 0

                                ; ***************************************************************************
                                ;NUMBER INPUT DIGIT FLAG
                                ; ***************************************************************************

030510  000000  000000          TTNBRF: 0       ;-1 IF ANY DIGIT TYPED

                                ; ***************************************************************************
                                ;KL10 & KI10 "INHPAG" SWITCH PAGING PREVENTION
                                ; ***************************************************************************

030511  000000  000000          PVPAGI: 0       ;IF NON-ZERO, OVERRIDE "INHPAG" SWITCH ACTION

                                ; ***************************************************************************
                                ;ERROR REPORTING ROUTINE ADDITIONAL USERS CONTROL INSTRUCTIONS
                                ; ***************************************************************************

030512  000000  000000          %ERHI1: 0       ;IF NON-ZERO, XCT'D AT START OF %ERUUO
030513  000000  000000          %ERHI2: 0       ;IF NON-ZERO, XCT'D AT END OF %ERUUO
030514  000000  000000          %ERHI3: 0       ;IF NON-ZERO, XCT'D AFTER "PC" OF %ERUUO

                                ; ***************************************************************************
                                ;SPECIAL USERS UUO INTERCEPT INSTRUCTION
                                ; ***************************************************************************

030515  000000  000000          $$UUO:  0       ;IF NON-ZERO, XCT'D AT START OF $UORTN
```

```
                                ; ************************************************************************
                                ;USER MODE MONITOR TYPE FLAG
                                ; ************************************************************************

030516  000000  000000          MONTYP: 0       ;0 = TOPS10, -1 = TOPS20

                                ; ************************************************************************
                                ;SPECIAL USERS MUUO INTERCEPT INSTRUCTION
                                ; ************************************************************************

030517  000000  000000          $$MUUO: 0       ;IF NON-ZERO, XCT'D AT START OF MUUOER

                                ; ************************************************************************
                                ;SPECIAL USERS USER MODE OUTPUT ERROR INTERCEPT INSTUCTION
                                ; ************************************************************************

030520  000000  000000          $$OUTER:0       ;IF NON-ZERO, XCT'D AT END OF USER MODE ERROR

                                ; ************************************************************************
                                ;"SWITCH" CALL USAGE CONTROL
                                ; ************************************************************************

030521  000000  000000          $$TOGGLE:0      ;IF NON-ZERO, USE C(CONSW) FOR SWITCHES

                                ; ************************************************************************
                                ;SPECIAL USERS ALTMODE SWITCH CALL INTERCEPT INSTRUCTIONS
                                ; ************************************************************************

030522  000000  000000          $$TAX1: 0       ;IF NON-ZERO, XCT'D AT START OF ALTMODE SWITCH CALL
030523  000000  000000          $$TAX2: 0       ;IF NON-ZERO, XCT'D AT END OF ALTMODE SWITCH CALL

                                ; ************************************************************************
                                ;SM10 (KS-10) PROCESSOR TYPE FLAG
                                ; ************************************************************************

030524  000000  000000          SM10:   0       ;IF -1 THIS IS A KS-10

                                ; ************************************************************************
                                ;RIGHT HALF SWITCHES PROMPT TABLE ADDRESS
                                ; ************************************************************************

030525  000000  000000          SWPTAB: 0       ;0 = NO PROMPT, ADR = ADR OF SIXBIT PROMPT TABLE

                                ; ************************************************************************
                                ;SPECIAL FUTURE EXPANSION ROOM
                                ; ************************************************************************

                                ; ************************************************************************
                                ;END OF FIXED STORAGE
                                ; ************************************************************************

030577                          LOC     30577
030577  000000  000000          ENDFIX: 0               ;END OF FIXED STORAGE
```

```
                                SUBTTL  *SPCCPU* SPECIAL BASIC CPU PROCESSOR CONTROL, 26-FEB-76

                                ;NEW DEFINITIONS USED BY THE KL10 SUBROUTINE PACKAGE

000000                          AC0=    0
030000                          DIAGNOS=30000           ;PDP-10 DIAGNOSTIC START ADDRESS
010000                          DDT=    10000           ;PDP-10 DDT START ADDRESS
020000                          DIAMON= 20000           ;PDP-10 DIAMON LOADER START ADDRESS
020000                          DONG11= 1B22            ;11 DOORBELL (FROM THE 10)

                                ;DTE20 DEVICE CODES

000200                          DTE==   200             ;DTE0
000204                          DTE0==  204
000204                          DTE1==  204
000210                          DTE2==  210
000214                          DTE3==  214

                                ;KL10 EPT COMMUNICATION AREA

000440                          $STD=   440             ;PDP-10 DIAGNOSTIC START ADDRESS
000441                          $DDT=   441             ;PDP-10 DDT START ADDRESS
000442                          $STL=   442             ;PDP-10 LOADER START ADDRESS
000443                          $STM=   443             ;PDP-10 MONITOR START ADDRESS

000444                          $DTFLG= 444             ;DTE20 OPERATION COMPLETE FLAG
000445                          $DTCLK= 445             ;DTE20 CLOCK INTERRUPT FLAG
000446                          $DTCI=  446             ;DTE20 CLOCK INTERRUPT INSTRUCTION
000447                          $DTT11= 447             ;DTE20 10 TO 11 ARGUMENT
000450                          $DTF11= 450             ;DTE20 11 TO 10 ARGUMENT
000451                          $DTCMD= 451             ;DTE20 TO 11 COMMAND WORD
000452                          $DTSEQ= 452             ;DTE20 OPERATION SEQUENCE NUMBER
000453                          $DTOPR= 453             ;DTE20 OPERATIONAL DTE #
000454                          $DTCHR= 454             ;DTE20 LAST TYPED CHARACTER
000455                          $DTMTD= 455             ;DTE20 MONITOR TTY OUTPUT COMPLETE FLAG
000456                          $DTMTI= 456             ;DTE20 MONITOR TTY INPUT FLAG

000457                          $DTSWR= 457             ;DTE20 CONSOLE SWITCH REGISTER
```

```
                                        ;SPECIAL "FIXED" REASSIGNMENTS

                    030600                      $$LOC=.                 ;SAVE CURRENT LOCATION

030000                                          LOC     30000
030000  254 00 0 00 030600          $$BEGIN:JRST    $$START             ;SETUP SPECIAL START
030001  254 00 0 00 030600                  JRST    $$START             ;"DIAMON" CHAIN START ADDRESS

000440                                          LOC     440
000440  254 00 0 00 030000          $STD:   JRST    BEGIN               ;SETUP FOR "STD"
000443                                          LOC     443
000443  254 00 0 00 030626          $STM:   JRST    $SPEC               ;SIMPLE RUN CONTROL

030057                                          LOC     30057
030057  254 00 0 00 030631          $BEGEND:JRST    $SPBEND             ;SETUP SPECIAL "BEGEND"

                                        ;SPECIAL MUUO, TRAP & PAGE FAIL SETUP

000420                                          LOC     420
000420  254 04 0 00 000420          $$420:  HALT    .                   ;KI10 PAGE FAIL
000421  255 00 0 00 000000          $$421:  JFCL                        ;OVERFLOW
000422  254 04 0 00 000422          $$422:  HALT    .                   ;PUSHDOWN OVERFLOW
000423  254 04 0 00 000423          $$423:  HALT    .                   ;TRAP 3
000424  000000  000000              $$424:  0                           ;MMUO
000425  000000  000000              $$425:  0                           ;MMUO PC
000426  000000  000000              $$426:  0                           ;KI10-PAGE FAIL, KL10-PROCESS CONTEXT
000427  254 04 0 00 000427          $$427:  HALT    .
000430  000000  000427              $$430:  427                         ;MMUO NEW PC'S
000431  000000  000427              $$431:  427
000432  000000  000427              $$432:  427
000433  000000  000427              $$433:  427
000434  000000  000427              $$434:  427
000435  000000  000427              $$435:  427
000436  000000  000427              $$436:  427
000437  000000  000427              $$437:  427

000500                                          LOC     500
000500  000000  000000              $$500:  0                           ;KL10 PAGE FAIL WORD
000501  000000  000000              $$501:  0                           ;KL10 PAGE FAIL PC
000502  000000  000503              $$502:  503                         ;KL10 PAGE FAIL NEW PC
000503  254 04 0 00 000503          $$503:  HALT    .
```

```
   768  030600                              LOC     $$LOC           ;RESET CURRENT LOCATION
   769
   770                                 ;SPECIAL STARTUP SEQUENCE
   771
   772  030600  402 00 0 00 030037     $$START:SETZM   USER
   773  030601  265 00 0 00 030602             JSP     0,.+1           ;IN USER MODE ?
   774  030602  603 00 0 00 010000             TLNE    0,USERF
   775  030603  476 00 0 00 030037             SETOM   USER            ;YES, SET CONTROL WORD
   776  030604  336 00 0 00 030042             SKIPN   MONFLG          ;SPECIAL USER MODE ?
   777  030605  402 00 0 00 030037             SETZM   USER            ;YES, RUN AS EXEC
   778  030606  332 00 0 00 030037             SKIPE   USER
   779  030607  254 00 0 00 030674             JRST    START           ;USER MODE, DON'T NEED CPU TYPE
   780
   781  030610  336 00 0 00 030044     $STKIL: SKIPN   MONTEN          ;LOADED BY 'DIAMON'' ?
   782  030611  476 00 0 00 030024             SETOM   ITRCNT          ;NO, RUN FOREVER
   783  030612  402 00 0 00 030041             SETZM   KLFLG           ;ASSUME KI10
   784  030613  200 01 0 00 034417             MOVE    1,[1,,1]
   785  030614  251 01 0 00 000001             BLT     1,1             ;HOPE THIS WORKS
   786  030615  316 01 0 00 034417             CAMN    1,[1,,1]        ;IF AC NE 1,,1 AFTER BLT, KL10
   787  030616  254 00 0 00 030674             JRST    START           ;KI10, NO ADDITIONAL SETUP
   788
   789  030617  402 00 0 00 000444     $STKL:  SETZM   $DTFLG
   790  030620  402 00 0 00 000445             SETZM   $DTCLK
   791  030621  200 00 0 00 000453             MOVE    $DTOPR          ;GET DTE #
   792  030622  436 00 0 00 030657             ORM     $$DTE0          ;INSERT IN DTE I/O INSTS
   793  030623  436 00 0 00 030670             ORM     $$DTE2
   794  030624  476 00 0 00 030041             SETOM   KLFLG           ;SET KL10 CONTROL FLAG
   795  030625  254 00 0 00 030674             JRST    START
   796
   797  030626  200 00 0 00 034420     $SPEC:  MOVE    [JRST STARTA]   ;SIMPLE RUN CONTROL
   798  030627  202 00 0 00 030633             MOVEM   $SPB1
   799  030630  254 00 0 00 030674             JRST    START
```

```
                                        ;SPECIAL "BEGEND" ROUTINE

030631  350 00 0 00 030047      $SPBEND:AOS     PASCNT          ;INCREMENT PASS COUNT
030632  370 00 0 00 030024              SOS     ITRCNT          ;DECREMENT ITERATION COUNT
030633  336 00 0 00 030037      $SPB1:  SKIPN   USER
030634  254 00 0 00 030642              JRST    $SPBEX          ;EXEC MODE

030635  332 00 0 00 030024      $SPBUS: SKIPE   ITRCNT          ;USER MODE, COMPLETED ?
030636  254 00 0 00 030715              JRST    STARTA          ;NO, KEEP RUNNING
030637  336 00 0 00 030044              SKIPN   MONTEN          ;DONE, LOADED BY "DIAMON" ?
030640  047 00 0 00 000012              EXIT                    ;NO, RETURN TO MONITOR
030641  254 00 1 00 030012              JRST    @RETURN         ;YES, RETURN TO "DIAMON"

030642  332 00 0 00 030041      $SPBEX: SKIPE   KLFLG
030643  254 00 0 00 030650              JRST    $SPBKL          ;KL10 & EXEC
030644  7 004 14 0 00 030024            DATAO   PI,ITRCNT       ;KI10 & EXEC, DISPLAY ITER COUNT
030645  332 00 0 00 030024              SKIPE   ITRCNT
030646  254 00 0 00 030715              JRST    STARTA          ;NOT COMPLETED YET
030647  254 00 1 00 030012              JRST    @RETURN         ;DONE

030650  336 00 0 00 030024      $SPBKL: SKIPN   ITRCNT
030651  254 00 0 00 030663              JRST    $SPKLD          ;KL10, EXEC & COMPLETED

030652  335 00 0 00 030043              SKIPGE  MONCTL
030653  254 00 0 00 030715              JRST    STARTA          ;"DIAMON" CONTROL
030654  201 00 0 00 000404              MOVEI   0,404           ;NOTIFY PDP-11 OF END OF PASS
030655  202 00 0 00 000451              MOVEM   0,$DTCMD
030656  402 00 0 00 000444              SETZM   $DTFLG
030657  7 200 20 0 00 020000    $$DTE0: CONO    DTE,DONG11
030660  336 00 0 00 000444              SKIPN   $DTFLG          ;WAIT TILL 11 RESPONDS
030661  254 00 0 00 030660              JRST    .-1
030662  254 00 0 00 030715              JRST    STARTA          ;KEEP RUNNING

                                        ;SPECIAL KL10 COMPLETED ROUTINE

030663  332 00 0 00 030044      $SPKLD: SKIPE   MONTEN
030664  254 00 1 00 030012              JRST    @RETURN         ;LOADED BY "DIAMON"

030665  201 00 0 00 000403              MOVEI   0,403           ;NOTIFY PDP-11 OF COMPLETION
030666  202 00 0 00 000451              MOVEM   0,$DTCMD
030667  402 00 0 00 000444              SETZM   $DTFLG
030670  7 200 20 0 00 020000    $$DTE2: CONO    DTE,DONG11
030671  336 00 0 00 000444              SKIPN   $DTFLG          ;SHOULD NEVER HAPPEN
030672  254 00 0 00 030671              JRST    .-1             ;11 NEVER RETURNS ON END OF PROGRAM
030673  254 04 0 00 030000              HALT    BEGIN           ;IF IT DOES, HALT.
```

```
                                        SUBTTL  DIAGNOSTIC SECTION

030674  336 00 0 00 030037      START:  SKIPN   USER
030675  254 00 0 00 030715              JRST    STARTA

030676  331 00 0 00 030043              SKIPL   MONCTL
030677  051 03 0 00 030701              TTCALL  3,PGMNAM
030700  254 00 0 00 030715              JRST    STARTA

030701                                  PGMNAM: ASCIZ/
030701  015 012 104 105 103            DECSYSTEM KL10 BASIC INSTRUCTION DIAGNOSTIC (4) [DFKAD]
030702  123 131 123 124 105
030703  115 040 113 114 061
030704  060 040 102 101 123
030705  111 103 040 111 116
030706  123 124 122 125 103
030707  124 111 117 116 040
030710  104 111 101 107 116
030711  117 123 124 111 103
030712  040 050 064 051 040
030713  133 104 106 113 101
030714  104 135 015 012 000     /

                                        ;Basic instruction test (4)
                                        ;The test is designed for initial debugging of
                                        ;Processor hardware and to detect (solid) failures
                                        ;In the field.

030715  254 00 0 00 030716      STARTA: JRST    .+1
```

```
                                SUBTTL  TEST OF AC HARDWARE AND INDEX REGISTERS

                                ;Note:  an "*" in the comment field of an instruction indicates that
                                ;       It is the tested instruction.


                                ;**********

                                ;This test verifies that ac1 is accessable
                                ;In this case, ac1 is preloaded with 1.
                                ;C(ac1) is then checked for a1.  this test passes if c(ac1)=1.
                                ;If this test fails, the ac hardware is probably faulty.

030716  400 00 0 00 00000C      C100:   SETZ                            ;Clear ac0
030717  201 01 0 00 000001               MOVEI   1,1                    ;Preload ac1 umth 1
030720  302 01 0 00 000001               CAIE    1,1                    ;Pass if c(ac1)=1
                                        STOP^
030721  254 04 0 00 030722               HALT    .+1                    ;Test failed if program halts here
030722  324 00 0 00 030723               JUMPA   .+1                    ;To loop change this instruction^

                                ;**********

                                ;This test verifies that ac2 is accessable.
                                ;In this case, ac2 is preloaded with 2.
                                ;C(ac2) is then checked for 2.  this test passes if c(ac2)=2.
                                ;If this test fails, the ac hardware is probably faulty.

030723  201 02 0 00 000002      C200:   MOVEI   2,2                    ;Preload ac2 with 2
030724  302 02 0 00 000002               CAIE    2,2                    ;Pass if c(ac2)=2
                                        STOP^
030725  254 04 0 00 030726               HALT    .+1                    ;Test failed if program halts here
030726  324 00 0 00 030727               JUMPA   .+1                    ;To loop charge this instruction^

                                ;**********
```

```
                                ;This test verifies that ac4 is accessable.
                                ;In this case, ac4 is preloaded with 4.
                                ;C(ac4) is then checked for 4.  this test passes if c(ac4)=4.
                                ;If this test fails, the ac hardware is probably faulty.

030727  201 04 0 00 000004      C300:   MOVEI   4,4             ;Preload ac4 with 4
030730  302 04 0 00 000004              CAIE    4,4             ;Pass if c(ac4)=4
                                        STOP^
030731  254 04 0 00 030732              HALT    .+1             ;Test failed if program halts here
030732  324 00 0 00 030733              JUMPA   .+1             ;To loop change this instruction^

                                ;**********

                                ;This test verifies that ac10 is accessable.
                                ;In this case, ac10 is preloaded with 10.
                                ;C(ac10) is then checked for 10.  this test passes if c(ac10)=10.
                                ;If this test fails, the ac hardware is probably faulty.

030733  201 10 0 00 000010      C400:   MOVEI   10,10           ;Preload ac10 with 10
030734  302 10 0 00 000010              CAIE    10,10           ;Pass if c(ac10)=10
                                        STOP^
030735  254 04 0 00 030736              HALT    .+1             ;Test failed if program halts here
030736  324 00 0 00 030737              JUMPA   .+1             ;To loop change this instruction^

                                ;**********
```

```
                                000500          SN=500
                        777777  777777                  ZZ=-1

                                                ;This test  verifies that all acs exist.  first, each ac is
                                                ;Preloaded with its own address. then, the contents of each
                                                ;Ac is checked for its address.  if any ac does not contain
                                                ;Its address, this test fails.

                                                C500:   REPEAT  ^D16,
                                                <               ZZ=ZZ+1
                                                                MOVEI   ZZ,ZZ                   ;Preload each ac with its addr>
                                    000000                      ZZ=ZZ+1
030737  201 00 0 00 000000                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000001                      ZZ=ZZ+1
030740  201 01 0 00 000001                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000002                      ZZ=ZZ+1
030741  201 02 0 00 000002                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000003                      ZZ=ZZ+1
030742  201 03 0 00 000003                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000004                      ZZ=ZZ+1
030743  201 04 0 00 000004                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000005                      ZZ=ZZ+1
030744  201 05 0 00 000005                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000006                      ZZ=ZZ+1
030745  201 06 0 00 000006                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000007                      ZZ=ZZ+1
030746  201 07 0 00 000007                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000010                      ZZ=ZZ+1
030747  201 10 0 00 000010                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000011                      ZZ=ZZ+1
030750  201 11 0 00 000011                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000012                      ZZ=ZZ+1
030751  201 12 0 00 000012                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000013                      ZZ=ZZ+1
030752  201 13 0 00 000013                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000014                      ZZ=ZZ+1
030753  201 14 0 00 000014                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000015                      ZZ=ZZ+1
030754  201 15 0 00 000015                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000016                      ZZ=ZZ+1
030755  201 16 0 00 000016                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr
                                    000017                      ZZ=ZZ+1
030756  201 17 0 00 000017                                      MOVEI   ZZ,ZZ                   ;Preload each ac with its addr

                                PAGE
```

```
                    000020              ZZ=20

                                        REPEAT  ^D16,<
                                    ;This test  verifies that all acs exist.  first, each ac is
                                    ;Preloaded with its own address. then, the contents of each
                                    ;Ac is checked for its address.  if any ac does not contain
                                    ;Its address, this test fails.

                                    SN=SN+1
                                            ZZ=ZZ-1
                                            CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
                                            STOP
                                                                            ;In case of failure, loop to "c500" address

                                    ;**********
                                    >

                                    ;This test  verifies that all acs exist.  first, each ac is
                                    ;Preloaded with its own address. then, the contents of each
                                    ;Ac is checked for its address.  if any ac does not contain
                                    ;Its address, this test fails.

                    000501          SN=SN+1
                    000017                  ZZ=ZZ-1
030757  302 17 0 00 000017                  CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
                                            STOP^
030760  254 04 0 00 030761                  HALT    .+1                     ;Test failed if program halts here
030761  324 00 0 00 030762                  JUMPA   .+1                     ;To loop change this instruction^
                                                                            ;In case of failure, loop to "c500" address

                                    ;**********


                                    ;This test  verifies that all acs exist.  first, each ac is
                                    ;Preloaded with its own address. then, the contents of each
                                    ;Ac is checked for its address.  if any ac does not contain
                                    ;Its address, this test fails.

                    000502          SN=SN+1
                    000016                  ZZ=ZZ-1
030762  302 16 0 00 000016                  CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
                                            STOP^
030763  254 04 0 00 030764                  HALT    .+1                     ;Test failed if program halts here
030764  324 00 0 00 030765                  JUMPA   .+1                     ;To loop change this instruction^
                                                                            ;In case of failure, loop to "c500" address

                                    ;**********


                                    ;This test  verifies that all acs exist.  first, each ac is
                                    ;Preloaded with its own address. then, the contents of each
                                    ;Ac is checked for its address.  if any ac does not contain
                                    ;Its address, this test fails.

                    000503          SN=SN+1
```

```
1032                    000015                  ZZ=ZZ-1
1033  030765  302 15 0 00 000015                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1034                                            STOP^
1035  030766  254 04 0 00 030767                HALT    .+1                     ;Test failed if program halts here
1036  030767  324 00 0 00 030770                JUMPA   .+1                     ;To loop change this instruction^
1037                                                                            ;In case of failure, loop to "c500" address
1038
1039                                    ;**********
1040
1041
1042                                    ;This test  verifies that all acs exist.  first, each ac is
1043                                    ;Preloaded with its own address. then, the contents of each
1044                                    ;Ac is checked for its address.  if any ac does not contain
1045                                    ;Its address, this test fails.
1046
1047                    000504          SN=SN+1
1048                    000014                  ZZ=ZZ-1
1049  030770  302 14 0 00 000014                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1050                                            STOP^
1051  030771  254 04 0 00 030772                HALT    .+1                     ;Test failed if program halts here
1052  030772  324 00 0 00 030773                JUMPA   .+1                     ;To loop change this instruction^
1053                                                                            ;In case of failure, loop to "c500" address
1054
1055                                    ;**********
1056
1057
1058                                    ;This test  verifies that all acs exist.  first, each ac is
1059                                    ;Preloaded with its own address. then, the contents of each
1060                                    ;Ac is checked for its address.  if any ac does not contain
1061                                    ;Its address, this test fails.
1062
1063                    000505          SN=SN+1
1064                    000013                  ZZ=ZZ-1
1065  030773  302 13 0 00 000013                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1066                                            STOP^
1067  030774  254 04 0 00 030775                HALT    .+1                     ;Test failed if program halts here
1068  030775  324 00 0 00 030776                JUMPA   .+1                     ;To loop change this instruction^
1069                                                                            ;In case of failure, loop to "c500" address
1070
1071                                    ;**********
1072
1073
1074                                    ;This test  verifies that all acs exist.  first, each ac is
1075                                    ;Preloaded with its own address. then, the contents of each
1076                                    ;Ac is checked for its address.  if any ac does not contain
1077                                    ;Its address, this test fails.
1078
1079                    000506          SN=SN+1
1080                    000012                  ZZ=ZZ-1
1081  030776  302 12 0 00 000012                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1082                                            STOP^
1083  030777  254 04 0 00 031000                HALT    .+1                     ;Test failed if program halts here
1084  031000  324 00 0 00 031001                JUMPA   .+1                     ;To loop change this instruction^
1085                                                                            ;In case of failure, loop to "c500" address
1086
```

```
1087                                    ;**********
1088
1089
1090                                    ;This test  verifies that all acs exist.  first, each ac is
1091                                    ;Preloaded with its own address. then, the contents of each
1092                                    ;Ac is checked for its address.  if any ac does not contain
1093                                    ;Its address, this test fails.
1094
1095                      000507        SN=SN+1
1096                      000011                ZZ=ZZ-1
1097  031001  302 11 0 00 000011                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1098                                            STOP^
1099  031002  254 04 0 00 031003                HALT    .+1                     ;Test failed if program halts here
1100  031003  324 00 0 00 031004                JUMPA   .+1                     ;To loop change this instruction^
1101                                                                            ;In case of failure, loop to "c500" address
1102
1103                                    ;**********
1104
1105
1106                                    ;This test  verifies that all acs exist.  first, each ac is
1107                                    ;Preloaded with its own address. then, the contents of each
1108                                    ;Ac is checked for its address.  if any ac does not contain
1109                                    ;Its address, this test fails.
1110
1111                      000510        SN=SN+1
1112                      000010                ZZ=ZZ-1
1113  031004  302 10 0 00 000010                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1114                                            STOP^
1115  031005  254 04 0 00 031006                HALT    .+1                     ;Test failed if program halts here
1116  031006  324 00 0 00 031007                JUMPA   .+1                     ;To loop change this instruction^
1117                                                                            ;In case of failure, loop to "c500" address
1118
1119                                    ;**********
1120
1121
1122                                    ;This test  verifies that all acs exist.  first, each ac is
1123                                    ;Preloaded with its own address. then, the contents of each
1124                                    ;Ac is checked for its address.  if any ac does not contain
1125                                    ;Its address, this test fails.
1126
1127                      000511        SN=SN+1
1128                      000007                ZZ=ZZ-1
1129  031007  302 07 0 00 000007                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1130                                            STOP^
1131  031010  254 04 0 00 031011                HALT    .+1                     ;Test failed if program halts here
1132  031011  324 00 0 00 031012                JUMPA   .+1                     ;To loop change this instruction^
1133                                                                            ;In case of failure, loop to "c500" address
1134
1135                                    ;**********
1136
1137
1138                                    ;This test  verifies that all acs exist.  first, each ac is
1139                                    ;Preloaded with its own address. then, the contents of each
1140                                    ;Ac is checked for its address.  if any ac does not contain
1141                                    ;Its address, this test fails.
```

```
1142
1143                    000512          SN=SN+1
1144                    000006                  ZZ=ZZ-1
1145  031012  302 06 0 00 000006               CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1146                                            STOP^
1147  031013  254 04 0 00 031014               HALT    .+1                     ;Test failed if program halts here
1148  031014  324 00 0 00 031015               JUMPA   .+1                     ;To loop change this instruction^
1149                                                                            ;In case of failure, loop to "c500" address
1150
1151                            ;**********
1152
1153
1154                            ;This test  verifies that all acs exist.  first, each ac is
1155                            ;Preloaded with its own address. then, the contents of each
1156                            ;Ac is checked for its address.  if any ac does not contain
1157                            ;Its address, this test fails.
1158
1159                    000513          SN=SN+1
1160                    000005                  ZZ=ZZ-1
1161  031015  302 05 0 00 000005               CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1162                                            STOP^
1163  031016  254 04 0 00 031017               HALT    .+1                     ;Test failed if program halts here
1164  031017  324 00 0 00 031020               JUMPA   .+1                     ;To loop change this instruction^
1165                                                                            ;In case of failure, loop to "c500" address
1166
1167                            ;**********
1168
1169
1170                            ;This test  verifies that all acs exist.  first, each ac is
1171                            ;Preloaded with its own address. then, the contents of each
1172                            ;Ac is checked for its address.  if any ac does not contain
1173                            ;Its address, this test fails.
1174
1175                    000514          SN=SN+1
1176                    000004                  ZZ=ZZ-1
1177  031020  302 04 0 00 000004               CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1178                                            STOP^
1179  031021  254 04 0 00 031022               HALT    .+1                     ;Test failed if program halts here
1180  031022  324 00 0 00 031023               JUMPA   .+1                     ;To loop change this instruction^
1181                                                                            ;In case of failure, loop to "c500" address
1182
1183                            ;**********
1184
1185
1186                            ;This test  verifies that all acs exist.  first, each ac is
1187                            ;Preloaded with its own address. then, the contents of each
1188                            ;Ac is checked for its address.  if any ac does not contain
1189                            ;Its address, this test fails.
1190
1191                    000515          SN=SN+1
1192                    000003                  ZZ=ZZ-1
1193  031023  302 03 0 00 000003               CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1194                                            STOP^
1195  031024  254 04 0 00 031025               HALT    .+1                     ;Test failed if program halts here
1196  031025  324 00 0 00 031026               JUMPA   .+1                     ;To loop change this instruction^
```

```
1197                                                                    ;In case of failure, loop to "c500" address
1198
1199                                    ;**********
1200
1201
1202                                    ;This test  verifies that all acs exist.  first, each ac is
1203                                    ;Preloaded with its own address. then, the contents of each
1204                                    ;Ac is checked for its address.  if any ac does not contain
1205                                    ;Its address, this test fails.
1206
1207                    000516          SN=SN+1
1208                    000002                  ZZ=ZZ-1
1209  031026  302 02 0 00 000002                 CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1210                                            STOP^
1211  031027  254 04 0 00 031030                 HALT    .+1                     ;Test failed if program halts here
1212  031030  324 00 0 00 031031                 JUMPA   .+1                     ;To loop change this instruction^
1213                                                                            ;In case of failure, loop to "c500" address
1214
1215                                    ;**********
1216
1217
1218                                    ;This test  verifies that all acs exist.  first, each ac is
1219                                    ;Preloaded with its own address. then, the contents of each
1220                                    ;Ac is checked for its address.  if any ac does not contain
1221                                    ;Its address, this test fails.
1222
1223                    000517          SN=SN+1
1224                    000001                  ZZ=ZZ-1
1225  031031  302 01 0 00 000001                 CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1226                                            STOP^
1227  031032  254 04 0 00 031033                 HALT    .+1                     ;Test failed if program halts here
1228  031033  324 00 0 00 031034                 JUMPA   .+1                     ;To loop change this instruction^
1229                                                                            ;In case of failure, loop to "c500" address
1230
1231                                    ;**********
1232
1233
1234                                    ;This test  verifies that all acs exist.  first, each ac is
1235                                    ;Preloaded with its own address. then, the contents of each
1236                                    ;Ac is checked for its address.  if any ac does not contain
1237                                    ;Its address, this test fails.
1238
1239                    000520          SN=SN+1
1240                    000000                  ZZ=ZZ-1
1241  031034  302 00 0 00 000000                 CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1242                                            STOP^
1243  031035  254 04 0 00 031036                 HALT    .+1                     ;Test failed if program halts here
1244  031036  324 00 0 00 031037                 JUMPA   .+1                     ;To loop change this instruction^
1245                                                                            ;In case of failure, loop to "c500" address
1246
1247                                    ;**********
1248
```

```
1249                    000600          SN=600
1250                    000000                  ZZ=0
1251
1252                                    ;This test verifies that an address within the ac range may be accessed as
1253                                    ;Either an ac or a memory location and that either reference refers to the
1254                                    ;Same hardware.  each ac is preloaded with its address.  then the contents
1255                                    ;Of each  ac  is  checked for its address.  if any ac does not contain its
1256                                    ;Address, this test fails.
1257
1258                                    C600:   REPEAT  ^D15,
1259                                    <       ZZ=ZZ+1
1260                                            MOVEI   ZZ                      ;Preload each ac with its address
1261                                            MOVEM   ZZ                      ;By referencing each ac as memory.>
1262                    000001                  ZZ=ZZ+1
1263  031037  201 00 0 00 000001                MOVEI   ZZ                      ;Preload each ac with its address
1264  031040  202 00 0 00 000001                MOVEM   ZZ                      ;By referencing each ac as memory.
1265                    000002                  ZZ=ZZ+1
1266  031041  201 00 0 00 000002                MOVEI   ZZ                      ;Preload each ac with its address
1267  031042  202 00 0 00 000002                MOVEM   ZZ                      ;By referencing each ac as memory.
1268                    000003                  ZZ=ZZ+1
1269  031043  201 00 0 00 000003                MOVEI   ZZ                      ;Preload each ac with its address
1270  031044  202 00 0 00 000003                MOVEM   ZZ                      ;By referencing each ac as memory.
1271                    000004                  ZZ=ZZ+1
1272  031045  201 00 0 00 000004                MOVEI   ZZ                      ;Preload each ac with its address
1273  031046  202 00 0 00 000004                MOVEM   ZZ                      ;By referencing each ac as memory.
1274                    000005                  ZZ=ZZ+1
1275  031047  201 00 0 00 000005                MOVEI   ZZ                      ;Preload each ac with its address
1276  031050  202 00 0 00 000005                MOVEM   ZZ                      ;By referencing each ac as memory.
1277                    000006                  ZZ=ZZ+1
1278  031051  201 00 0 00 000006                MOVEI   ZZ                      ;Preload each ac with its address
1279  031052  202 00 0 00 000006                MOVEM   ZZ                      ;By referencing each ac as memory.
1280                    000007                  ZZ=ZZ+1
1281  031053  201 00 0 00 000007                MOVEI   ZZ                      ;Preload each ac with its address
1282  031054  202 00 0 00 000007                MOVEM   ZZ                      ;By referencing each ac as memory.
1283                    000010                  ZZ=ZZ+1
1284  031055  201 00 0 00 000010                MOVEI   ZZ                      ;Preload each ac with its address
1285  031056  202 00 0 00 000010                MOVEM   ZZ                      ;By referencing each ac as memory.
1286                    000011                  ZZ=ZZ+1
1287  031057  201 00 0 00 000011                MOVEI   ZZ                      ;Preload each ac with its address
1288  031060  202 00 0 00 000011                MOVEM   ZZ                      ;By referencing each ac as memory.
1289                    000012                  ZZ=ZZ+1
1290  031061  201 00 0 00 000012                MOVEI   ZZ                      ;Preload each ac with its address
1291  031062  202 00 0 00 000012                MOVEM   ZZ                      ;By referencing each ac as memory.
1292                    000013                  ZZ=ZZ+1
1293  031063  201 00 0 00 000013                MOVEI   ZZ                      ;Preload each ac with its address
1294  031064  202 00 0 00 000013                MOVEM   ZZ                      ;By referencing each ac as memory.
1295                    000014                  ZZ=ZZ+1
1296  031065  201 00 0 00 000014                MOVEI   ZZ                      ;Preload each ac with its address
1297  031066  202 00 0 00 000014                MOVEM   ZZ                      ;By referencing each ac as memory.
1298                    000015                  ZZ=ZZ+1
1299  031067  201 00 0 00 000015                MOVEI   ZZ                      ;Preload each ac with its address
1300  031070  202 00 0 00 000015                MOVEM   ZZ                      ;By referencing each ac as memory.
1301                    000016                  ZZ=ZZ+1
1302  031071  201 00 0 00 000016                MOVEI   ZZ                      ;Preload each ac with its address
1303  031072  202 00 0 00 000016                MOVEM   ZZ                      ;By referencing each ac as memory.
```

```
                        000017                  ZZ=ZZ+1
031073  201 00 0 00 000017                      MOVEI   ZZ                      ;Preload each ac with its address
031074  202 00 0 00 000017                      MOVEM   ZZ                      ;By referencing each ac as memory.

                        000020                  ZZ=20

                                                REPEAT  ^D15,<
                                        ;This test verifies that an address within the ac range may be accessed as
                                        ;Either an ac or a memory location and that either reference refers to the
                                        ;Same hardware.  each ac is preloaded with its address.  then the contents
                                        ;Of each  ac  is  checked for its address.  if any ac does not contain its
                                        ;Address, this test fails.

                                        SN=SN+1
                                                ZZ=ZZ-1
                                                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
                                                STOP
                                                                                ;In case of failure, loop to "c600" address

                                        ;**********
                                        >

                                        ;This test verifies that an address within the ac range may be accessed as
                                        ;Either an ac or a memory location and that either reference refers to the
                                        ;Same hardware.  each ac is preloaded with its address.  then the contents
                                        ;Of each  ac  is  checked for its address.  if any ac does not contain its
                                        ;Address, this test fails.

                        000601          SN=SN+1
                        000017                  ZZ=ZZ-1
031075  302 17 0 00 000017                      CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
                                                STOP^
031076  254 04 0 00 031077                      HALT    .+1                     ;Test failed if program halts here
031077  324 00 0 00 031100                      JUMPA   .+1                     ;To loop change this instruction^
                                                                                ;In case of failure, loop to "c600" address

                                        ;**********


                                        ;This test verifies that an address within the ac range may be accessed as
                                        ;Either an ac or a memory location and that either reference refers to the
                                        ;Same hardware.  each ac is preloaded with its address.  then the contents
                                        ;Of each  ac  is  checked for its address.  if any ac does not contain its
                                        ;Address, this test fails.

                        000602          SN=SN+1
                        000016                  ZZ=ZZ-1
031100  302 16 0 00 000016                      CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
                                                STOP^
031101  254 04 0 00 031102                      HALT    .+1                     ;Test failed if program halts here
031102  324 00 0 00 031103                      JUMPA   .+1                     ;To loop change this instruction^
                                                                                ;In case of failure, loop to "c600" address

                                        ;**********
```

```
1359
1360                                    ;This test verifies that an address within the ac range may be accessed as
1361                                    ;Either an ac or a memory location and that either reference refers to the
1362                                    ;Same hardware.  each ac is preloaded with its address.  then the contents
1363                                    ;Of each  ac  is  checked for its address.  if any ac does not contain its
1364                                    ;Address, this test fails.
1365
1366                      000603        SN=SN+1
1367                      000015                ZZ=ZZ-1
1368  031103  302 15 0 00 000015                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1369                                            STOP^
1370  031104  254 04 0 00 031105                HALT    .+1                     ;Test failed if program halts here
1371  031105  324 00 0 00 031106                JUMPA   .+1                     ;To loop change this instruction^
1372                                                                            ;In case of failure, loop to "c600" address
1373
1374                                    ;**********
1375
1376
1377                                    ;This test verifies that an address within the ac range may be accessed as
1378                                    ;Either an ac or a memory location and that either reference refers to the
1379                                    ;Same hardware.  each ac is preloaded with its address.  then the contents
1380                                    ;Of each  ac  is  checked for its address.  if any ac does not contain its
1381                                    ;Address, this test fails.
1382
1383                      000604        SN=SN+1
1384                      000014                ZZ=ZZ-1
1385  031106  302 14 0 00 000014                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1386                                            STOP^
1387  031107  254 04 0 00 031110                HALT    .+1                     ;Test failed if program halts here
1388  031110  324 00 0 00 031111                JUMPA   .+1                     ;To loop change this instruction^
1389                                                                            ;In case of failure, loop to "c600" address
1390
1391                                    ;**********
1392
1393
1394                                    ;This test verifies that an address within the ac range may be accessed as
1395                                    ;Either an ac or a memory location and that either reference refers to the
1396                                    ;Same hardware.  each ac is preloaded with its address.  then the contents
1397                                    ;Of each  ac  is  checked for its address.  if any ac does not contain its
1398                                    ;Address, this test fails.
1399
1400                      000605        SN=SN+1
1401                      000013                ZZ=ZZ-1
1402  031111  302 13 0 00 000013                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1403                                            STOP^
1404  031112  254 04 0 00 031113                HALT    .+1                     ;Test failed if program halts here
1405  031113  324 00 0 00 031114                JUMPA   .+1                     ;To loop change this instruction^
1406                                                                            ;In case of failure, loop to "c600" address
1407
1408                                    ;**********
1409
1410
1411                                    ;This test verifies that an address within the ac range may be accessed as
1412                                    ;Either an ac or a memory location and that either reference refers to the
1413                                    ;Same hardware.  each ac is preloaded with its address.  then the contents
```

```
1414                                    ;Of each  ac  is  checked for its address.  if any ac does not contain its
1415                                    ;Address, this test fails.
1416
1417                    000606          SN=SN+1
1418                    000012                  ZZ=ZZ-1
1419  031114  302 12 0 00 000012                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1420                                            STOP^
1421  031115  254 04 0 00 031116                HALT    .+1                     ;Test failed if program halts here
1422  031116  324 00 0 00 031117                JUMPA   .+1                     ;To loop change this instruction^
1423                                                                            ;In case of failure, loop to "c600" address
1424
1425                                    ;**********
1426
1427
1428                                    ;This test verifies that an address within the ac range may be accessed as
1429                                    ;Either an ac or a memory location and that either reference refers to the
1430                                    ;Same hardware.  each ac is preloaded with its address.  then the contents
1431                                    ;Of each  ac  is  checked for its address.  if any ac does not contain its
1432                                    ;Address, this test fails.
1433
1434                    000607          SN=SN+1
1435                    000011                  ZZ=ZZ-1
1436  031117  302 11 0 00 000011                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1437                                            STOP^
1438  031120  254 04 0 00 031121                HALT    .+1                     ;Test failed if program halts here
1439  031121  324 00 0 00 031122                JUMPA   .+1                     ;To loop change tnis instruction^
1440                                                                            ;In case of failure, loop to "c600" address
1441
1442                                    ;**********
1443
1444
1445                                    ;This test verifies that an address within the ac range may be accessed as
1446                                    ;Either an ac or a memory location and that either reference refers to the
1447                                    ;Same hardware.  each ac is preloaded with its address.  then the contents
1448                                    ;Of each  ac  is  checked for its address.  if any ac does not contain its
1449                                    ;Address, this test fails.
1450
1451                    000610          SN=SN+1
1452                    000010                  ZZ=ZZ-1
1453  031122  302 10 0 00 000010                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1454                                            STOP^
1455  031123  254 04 0 00 031124                HALT    .+1                     ;Test failed if program halts here
1456  031124  324 00 0 00 031125                JUMPA   .+1                     ;To loop change this instruction^
1457                                                                            ;In case of failure, loop to "c600" address
1458
1459                                    ;**********
1460
1461
1462                                    ;This test verifies that an address within the ac range may be accessed as
1463                                    ;Either an ac or a memory location and that either reference refers to the
1464                                    ;Same hardware.  each ac is preloaded with its address.  then the contents
1465                                    ;Of each  ac  is  checked for its address.  if any ac does not contain its
1466                                    ;Address, this test fails.
1467
1468                    000611          SN=SN+1
```

```
1469                        000007              ZZ=ZZ-1
1470  031125  302 07 0 00 000007                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1471                                            STOP^
1472  031126  254 04 0 00 031127                HALT    .+1                     ;Test failed if program halts here
1473  031127  324 00 0 00 031130                JUMPA   .+1                     ;To loop change this instruction^
1474                                                                            ;In case of failure, loop to "c600" address
1475
1476                                    ;**********
1477
1478
1479                                    ;This test verifies that an address within the ac range may be accessed as
1480                                    ;Either an ac or a memory location and that either reference refers to the
1481                                    ;Same hardware.  each ac is preloaded with its address.  then the contents
1482                                    ;Of each  ac  is  checked for its address.  if any ac does not contain its
1483                                    ;Address, this test fails.
1484
1485                        000612      SN=SN+1
1486                        000006              ZZ=ZZ-1
1487  031130  302 06 0 00 000006                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1488                                            STOP^
1489  031131  254 04 0 00 031132                HALT    .+1                     ;Test failed if program halts here
1490  031132  324 00 0 00 031133                JUMPA   .+1                     ;To loop change this instruction^
1491                                                                            ;In case of failure, loop to "c600" address
1492
1493                                    ;**********
1494
1495
1496                                    ;This test verifies that an address within the ac range may be accessed as
1497                                    ;Either an ac or a memory location and that either reference refers to the
1498                                    ;Same hardware.  each ac is preloaded with its address.  then the contents
1499                                    ;Of each  ac  is  checked for its address.  if any ac does not contain its
1500                                    ;Address, this test fails.
1501
1502                        000613      SN=SN+1
1503                        000005              ZZ=ZZ-1
1504  031133  302 05 0 00 000005                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1505                                            STOP^
1506  031134  254 04 0 00 031135                HALT    .+1                     ;Test failed if program halts here
1507  031135  324 00 0 00 031136                JUMPA   .+1                     ;To loop change this instruction^
1508                                                                            ;In case of failure, loop to "c600" address
1509
1510                                    ;**********
1511
1512
1513                                    ;This test verifies that an address within the ac range may be accessed as
1514                                    ;Either an ac or a memory location and that either reference refers to the
1515                                    ;Same hardware.  each ac is preloaded with its address.  then the contents
1516                                    ;Of each  ac  is  checked for its address.  if any ac does not contain its
1517                                    ;Address, this test fails.
1518
1519                        000614      SN=SN+1
1520                        000004              ZZ=ZZ-1
1521  031136  302 04 0 00 000004                CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1522                                            STOP^
1523  031137  254 04 0 00 031140                HALT    .+1                     ;Test failed if program halts here
```

```
1524  031140  324 00 0 00 031141          JUMPA   .+1                     ;To loop change this instruction^
1525                                                                      ;In case of failure, loop to "c600" address
1526
1527                                ;**********
1528
1529
1530                                ;This test verifies that an address within the ac range may be accessed as
1531                                ;Either an ac or a memory location and that either reference refers to the
1532                                ;Same hardware.  each ac is preloaded with its address.  then the contents
1533                                ;Of each  ac  is  checked for its address.  if any ac does not contain its
1534                                ;Address, this test fails.
1535
1536                    000615      SN=SN+1
1537                    000003              ZZ=ZZ-1
1538  031141  302 03 0 00 000003            CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1539                                        STOP^
1540  031142  254 04 0 00 031143            HALT    .+1                     ;Test failed if program halts here
1541  031143  324 00 0 00 031144            JUMPA   .+1                     ;To loop change this instruction^
1542                                                                      ;In case of failure, loop to "c600" address
1543
1544                                ;**********
1545
1546
1547                                ;This test verifies that an address within the ac range may be accessed as
1548                                ;Either an ac or a memory location and that either reference refers to the
1549                                ;Same hardware.  each ac is preloaded with its address.  then the contents
1550                                ;Of each  ac  is  checked for its address.  if any ac does not contain its
1551                                ;Address, this test fails.
1552
1553                    000616      SN=SN+1
1554                    000002              ZZ=ZZ-1
1555  031144  302 02 0 00 000002            CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1556                                        STOP^
1557  031145  254 04 0 00 031146            HALT    .+1                     ;Test failed if program halts here
1558  031146  324 00 0 00 031147            JUMPA   .+1                     ;To loop change this instruction^
1559                                                                      ;In case of failure, loop to "c600" address
1560
1561                                ;**********
1562
1563
1564                                ;This test verifies that an address within the ac range may be accessed as
1565                                ;Either an ac or a memory location and that either reference refers to the
1566                                ;Same hardware.  each ac is preloaded with its address.  then the contents
1567                                ;Of each  ac  is  checked for its address.  if any ac does not contain its
1568                                ;Address, this test fails.
1569
1570                    000617      SN=SN+1
1571                    000001              ZZ=ZZ-1
1572  031147  302 01 0 00 000001            CAIE    ZZ,ZZ                   ;Check that each ac contains its own address
1573                                        STOP^
1574  031150  254 04 0 00 031151            HALT    .+1                     ;Test failed if program halts here
1575  031151  324 00 0 00 031152            JUMPA   .+1                     ;To loop change this instruction^
1576                                                                      ;In case of failure, loop to "c600" address
1577
1578                                ;**********
```

1579

```
1580                    000700          SN=700
1581                    000000                  ZZ=0
1582
1583                                    C700:   REPEAT  ^D15,<
1584                                    ;This test verifies that the index registers are accessable. first, an
1585                                    ;Index register is preloaded with its address.  then, it is referenced
1586                                    ;As an index register.  if it contains its address, this test  passes.
1587                                    ;This test is repeated 15 times in order to test each index register.
1588
1589                                    SN=SN+1
1590                                            ZZ=ZZ+1
1591                                            MOVEI   ZZ,ZZ                   ;Preload index register with its address
1592                                            CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
1593                                            STOP
1594
1595                                    ;**********
1596                                    >
1597
1598                                    ;This test verifies that the index registers are accessable. first, an
1599                                    ;Index register is preloaded with its address.  then, it is referenced
1600                                    ;As an index register.  if it contains its address, this test  passes.
1601                                    ;This test is repeated 15 times in order to test each index register.
1602
1603                    000701          SN=SN+1
1604                    000001                  ZZ=ZZ+1
1605  031152  201 01 0 00 000001                  MOVEI   ZZ,ZZ                   ;Preload index register with its address
1606  031153  302 01 0 01 000000                  CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
1607                                            STOP^
1608  031154  254 04 0 00 031155                  HALT    .+1                     ;Test failed if program halts here
1609  031155  324 00 0 00 031156                  JUMPA   .+1                     ;To loop change this instruction^
1610
1611                                    ;**********
1612
1613
1614                                    ;This test verifies that the index registers are accessable. first, an
1615                                    ;Index register is preloaded with its address.  then, it is referenced
1616                                    ;As an index register.  if it contains its address, this test  passes.
1617                                    ;This test is repeated 15 times in order to test each index register.
1618
1619                    000702          SN=SN+1
1620                    000002                  ZZ=ZZ+1
1621  031156  201 02 0 00 000002                  MOVEI   ZZ,ZZ                   ;Preload index register with its address
1622  031157  302 02 0 02 000000                  CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
1623                                            STOP^
1624  031160  254 04 0 00 031161                  HALT    .+1                     ;Test failed if program halts here
1625  031161  324 00 0 00 031162                  JUMPA   .+1                     ;To loop change this instruction^
1626
1627                                    ;**********
1628
1629
1630                                    ;This test verifies that the index registers are accessable. first, an
1631                                    ;Index register is preloaded with its address.  then, it is referenced
1632                                    ;As an index register.  if it contains its address, this test  passes.
1633                                    ;This test is repeated 15 times in order to test each index register.
1634
```

```
                        000703          SN=SN+1
                        000003                  ZZ=ZZ+1
031162  201 03 0 00 000003                      MOVEI   ZZ,ZZ                   ;Preload index register with its address
031163  302 03 0 03 000000                      CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
                                                STOP^
031164  254 04 0 00 031165                      HALT    .+1                     ;Test failed if program halts here
031165  324 00 0 00 031166                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test verifies that the index registers are accessable. first, an
                                        ;Index register is preloaded with its address.  then, it is referenced
                                        ;As an index register.  if it contains its address, this test  passes.
                                        ;This test is repeated 15 times in order to test each index register.

                        000704          SN=SN+1
                        000004                  ZZ=ZZ+1
031166  201 04 0 00 000004                      MOVEI   ZZ,ZZ                   ;Preload index register with its address
031167  302 04 0 04 000000                      CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
                                                STOP^
031170  254 04 0 00 031171                      HALT    .+1                     ;Test failed if program halts here
031171  324 00 0 00 031172                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test verifies that the index registers are accessable. first, an
                                        ;Index register is preloaded with its address.  then, it is referenced
                                        ;As an index register.  if it contains its address, this test  passes.
                                        ;This test is repeated 15 times in order to test each index register.

                        000705          SN=SN+1
                        000005                  ZZ=ZZ+1
031172  201 05 0 00 000005                      MOVEI   ZZ,ZZ                   ;Preload index register with its address
031173  302 05 0 05 000000                      CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
                                                STOP^
031174  254 04 0 00 031175                      HALT    .+1                     ;Test failed if program halts here
031175  324 00 0 00 031176                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test verifies that the index registers are accessable. first, an
                                        ;Index register is preloaded with its address.  then, it is referenced
                                        ;As an index register.  if it contains its address, this test  passes.
                                        ;This test is repeated 15 times in order to test each index register.

                        000706          SN=SN+1
                        000006                  ZZ=ZZ+1
031176  201 06 0 00 000006                      MOVEI   ZZ,ZZ                   ;Preload index register with its address
031177  302 06 0 06 000000                      CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
                                                STOP^
031200  254 04 0 00 031201                      HALT    .+1                     ;Test failed if program halts here
031201  324 00 0 00 031202                      JUMPA   .+1                     ;To loop change this instruction^
```

```
1690
1691                                    ;**********
1692
1693
1694                                    ;This test verifies that the index registers are accessable. first, an
1695                                    ;Index register is preloaded with its address.  then, it is referenced
1696                                    ;As an index register.  if it contains its address, this test  passes.
1697                                    ;This test is repeated 15 times in order to test each index register.
1698
1699                    000707          SN=SN+1
1700                    000007                  ZZ=ZZ+1
1701  031202  201 07 0 00 000007                MOVEI   ZZ,ZZ                   ;Preload index register with its address
1702  031203  302 07 0 07 000000                CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
1703                                            STOP^
1704  031204  254 04 0 00 031205                HALT    .+1                     ;Test failed if program halts here
1705  031205  324 00 0 00 031206                JUMPA   .+1                     ;To loop change this instruction^
1706
1707                                    ;**********
1708
1709
1710                                    ;This test verifies that the index registers are accessable. first, an
1711                                    ;Index register is preloaded with its address.  then, it is referenced
1712                                    ;As an index register.  if it contains its address, this test  passes.
1713                                    ;This test is repeated 15 times in order to test each index register.
1714
1715                    000710          SN=SN+1
1716                    000010                  ZZ=ZZ+1
1717  031206  201 10 0 00 000010                MOVEI   ZZ,ZZ                   ;Preload index register with its address
1718  031207  302 10 0 10 000000                CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
1719                                            STOP^
1720  031210  254 04 0 00 031211                HALT    .+1                     ;Test failed if program halts here
1721  031211  324 00 0 00 031212                JUMPA   .+1                     ;To loop change this instruction^
1722
1723                                    ;**********
1724
1725
1726                                    ;This test verifies that the index registers are accessable. first, an
1727                                    ;Index register is preloaded with its address.  then, it is referenced
1728                                    ;As an index register.  if it contains its address, this test  passes.
1729                                    ;This test is repeated 15 times in order to test each index register.
1730
1731                    000711          SN=SN+1
1732                    000011                  ZZ=ZZ+1
1733  031212  201 11 0 00 000011                MOVEI   ZZ,ZZ                   ;Preload index register with its address
1734  031213  302 11 0 11 000000                CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
1735                                            STOP^
1736  031214  254 04 0 00 031215                HALT    .+1                     ;Test failed if program halts here
1737  031215  324 00 0 00 031216                JUMPA   .+1                     :To loop change this instruction^
1738
1739                                    ;**********
1740
1741
1742                                    ;This test verifies that the index registers are accessable. first, an
1743                                    ;Index register is preloaded with its address.  then, it is referenced
1744                                    ;As an index register.  if it contains its address, this test  passes.
```

```
                                ;This test is repeated 15 times in order to test each index register.
                                
                    000712      SN=SN+1
                    000012              ZZ=ZZ+1
031216  201 12 0 00 000012              MOVEI   ZZ,ZZ               ;Preload index register with its address
031217  302 12 0 12 000000              CAIE    ZZ,(ZZ)             ;Pass if index register contains its address
                                        STOP^
031220  254 04 0 00 031221              HALT    .+1                 ;Test failed if program halts here
031221  324 00 0 00 031222              JUMPA   .+1                 ;To loop change this instruction^
                                
                                ;**********
                                
                                
                                ;This test verifies that the index registers are accessable. first, an
                                ;Index register is preloaded with its address.  then, it is referenced
                                ;As an index register.  if it contains its address, this test  passes.
                                ;This test is repeated 15 times in order to test each index register.
                                
                    000713      SN=SN+1
                    000013              ZZ=ZZ+1
031222  201 13 0 00 000013              MOVEI   ZZ,ZZ               ;Preload index register with its address
031223  302 13 0 13 000000              CAIE    ZZ,(ZZ)             ;Pass if index register contains its address
                                        STOP^
031224  254 04 0 00 031225              HALT    .+1                 ;Test failed if program halts here
031225  324 00 0 00 031226              JUMPA   .+1                 ;To loop change this instruction^
                                
                                ;**********
                                
                                
                                ;This test verifies that the index registers are accessable. first, an
                                ;Index register is preloaded with its address.  then, it is referenced
                                ;As an index register.  if it contains its address, this test  passes.
                                ;This test is repeated 15 times in order to test each index register.
                                
                    000714      SN=SN+1
                    000014              ZZ=ZZ+1
031226  201 14 0 00 000014              MOVEI   ZZ,ZZ               ;Preload index register with its address
031227  302 14 0 14 000000              CAIE    ZZ,(ZZ)             ;Pass if index register contains its address
                                        STOP^
031230  254 04 0 00 031231              HALT    .+1                 ;Test failed if program halts here
031231  324 00 0 00 031232              JUMPA   .+1                 ;To loop change this instruction^
                                
                                ;**********
                                
                                
                                ;This test verifies that the index registers are accessable. first, an
                                ;Index register is preloaded with its address.  then, it is referenced
                                ;As an index register.  if it contains its address, this test  passes.
                                ;This test is repeated 15 times in order to test each index register.
                                
                    000715      SN=SN+1
                    000015              ZZ=ZZ+1
031232  201 15 0 00 000015              MOVEI   ZZ,ZZ               ;Preload index register with its address
031233  302 15 0 15 000000              CAIE    ZZ,(ZZ)             ;Pass if index register contains its address
                                        STOP^
```

```
1800  031234  254 04 0 00 031235          HALT    .+1                     ;Test failed if program halts here
1801  031235  324 00 0 00 031236          JUMPA   .+1                     ;To loop change this instruction^
1802
1803                                  ;**********
1804
1805
1806                                  ;This test verifies that the index registers are accessable. first, an
1807                                  ;Index register is preloaded with its address.  then, it is referenced
1808                                  ;As an index register.  if it contains its address, this test  passes.
1809                                  ;This test is repeated 15 times in order to test each index register.
1810
1811                    000716        SN=SN+1
1812                    000016                ZZ=ZZ+1
1813  031236  201 16 0 00 000016          MOVEI   ZZ,ZZ                   ;Preload index register with its address
1814  031237  302 16 0 16 000000          CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
1815                                      STOP^
1816  031240  254 04 0 00 031241          HALT    .+1                     ;Test failed if program halts here
1817  031241  324 00 0 00 031242          JUMPA   .+1                     ;To loop change this instruction^
1818
1819                                  ;**********
1820
1821
1822                                  ;This test verifies that the index registers are accessable. first, an
1823                                  ;Index register is preloaded with its address.  then, it is referenced
1824                                  ;As an index register.  if it contains its address, this test  passes.
1825                                  ;This test is repeated 15 times in order to test each index register.
1826
1827                    000717        SN=SN+1
1828                    000017                ZZ=ZZ+1
1829  031242  201 17 0 00 000017          MOVEI   ZZ,ZZ                   ;Preload index register with its address
1830  031243  302 17 0 17 000000          CAIE    ZZ,(ZZ)                 ;Pass if index register contains its address
1831                                      STOP^
1832  031244  254 04 0 00 031245          HALT    .+1                     ;Test failed if program halts here
1833  031245  324 00 0 00 031246          JUMPA   .+1                     ;To loop change this instruction^
1834
1835                                  ;**********
1836
```

```
                        001000          SN=1000
                        000000                  ZZ=0

                                        C1000:  REPEAT  ^D15,<
                                        ;This test verifies that all index registers index correctly. first, both
                                        ;Halves of the index register are preloaded with its  address.  then, the
                                        ;Index register is referenced. it is then checked for its address in both
                                        ;Halves.  if it contains its address in both halves this test passes.
                                        ;This test is repeated 15 times in order to test each index register.

                                        SN=SN+1
                                                ZZ=ZZ+1
                                                MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
                                                HRLI    ZZ,ZZ                   ;Pass if index register contains its address
                                                CAME    ZZ,(ZZ)                 ;In both halves
                                                STOP

                                        ;**********
                                        >

                                        ;This test verifies that all index registers index correctly. first, both
                                        ;Halves of the index register are preloaded with its  address.  then, the
                                        ;Index register is referenced. it is then checked for its address in both
                                        ;Halves.  if it contains its address in both halves this test passes.
                                        ;This test is repeated 15 times in order to test each index register.

                        001001          SN=SN+1
                        000001                  ZZ=ZZ+1
031246  201 01 0 00 000001                      MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
031247  505 01 0 00 000001                      HRLI    ZZ,ZZ                   ;Pass if index register contains its address
031250  312 01 0 01 000000                      CAME    ZZ,(ZZ)                 ;In both halves
                                                STOP^
031251  254 04 0 00 031252                      HALT    .+1                     ;Test failed if program halts here
031252  324 00 0 00 031253                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test verifies that all index registers index correctly. first, both
                                        ;Halves of the index register are preloaded with its  address.  then, the
                                        ;Index register is referenced. it is then checked for its address in both
                                        ;Halves.  if it contains its address in both halves this test passes.
                                        ;This test is repeated 15 times in order to test each index register.

                        001002          SN=SN+1
                        000002                  ZZ=ZZ+1
031253  201 02 0 00 000002                      MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
031254  505 02 0 00 000002                      HRLI    ZZ,ZZ                   ;Pass if index register contains its address
031255  312 02 0 02 000000                      CAME    ZZ,(ZZ)                 ;In both halves
                                                STOP^
031256  254 04 0 00 031257                      HALT    .+1                     ;Test failed if program halts here
031257  324 00 0 00 031260                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                        ;This test verifies that all index registers index correctly. first, both
                                        ;Halves of the index register are preloaded with its  address.  then, the
                                        ;Index register is referenced. it is then checked for its address in both
                                        ;Halves.  if it contains its address in both halves this test passes.
                                        ;This test is repeated 15 times in order to test each index register.

                    001003              SN=SN+1
                    000003                      ZZ=ZZ+1
031260  201 03 0 00 000003                      MOVEI   ZZ,ZZ               ;Preload both halves of index reg with its address
031261  505 03 0 00 000003                      HRLI    ZZ,ZZ               ;Pass if index register contains its address
031262  312 03 0 03 000000                      CAME    ZZ,(ZZ)             ;In both halves
                                                STOP^
031263  254 04 0 00 031264                      HALT    .+1                 ;Test failed if program halts here
031264  324 00 0 00 031265                      JUMPA   .+1                 ;To loop change this instruction^

                                        ;**********


                                        ;This test verifies that all index registers index correctly. first, both
                                        ;Halves of the index register are preloaded with its  address.  then, the
                                        ;Index register is referenced. it is then checked for its address in both
                                        ;Halves.  if it contains its address in both halves this test passes.
                                        ;This test is repeated 15 times in order to test each index register.

                    001004              SN=SN+1
                    000004                      ZZ=ZZ+1
031265  201 04 0 00 000004                      MOVEI   ZZ,ZZ               ;Preload both halves of index reg with its address
031266  505 04 0 00 000004                      HRLI    ZZ,ZZ               ;Pass if index register contains its address
031267  312 04 0 04 000000                      CAME    ZZ,(ZZ)             ;In both halves
                                                STOP^
031270  254 04 0 00 031271                      HALT    .+1                 ;Test failed if program halts here
031271  324 00 0 00 031272                      JUMPA   .+1                 ;To loop change this instruction^

                                        ;**********


                                        ;This test verifies that all index registers index correctly. first, both
                                        ;Halves of the index register are preloaded with its  address.  then, the
                                        ;Index register is referenced. it is then checked for its address in both
                                        ;Halves.  if it contains its address in both halves this test passes.
                                        ;This test is repeated 15 times in order to test each index register.

                    001005              SN=SN+1
                    000005                      ZZ=ZZ+1
031272  201 05 0 00 000005                      MOVEI   ZZ,ZZ               ;Preload both halves of index reg with its address
031273  505 05 0 00 000005                      HRLI    ZZ,ZZ               ;Pass if index register contains its address
031274  312 05 0 05 000000                      CAME    ZZ,(ZZ)             ;In both halves
                                                STOP^
031275  254 04 0 00 031276                      HALT    .+1                 ;Test failed if program halts here
031276  324 00 0 00 031277                      JUMPA   .+1                 ;To loop change this instruction^

                                        ;**********
```

```
                                        ;This test verifies that all index registers index correctly. first, both
                                        ;Halves of the index register are preloaded with its  address.  then, the
                                        ;Index register is referenced. it is then checked for its address in both
                                        ;Halves.  if it contains its address in both halves this test passes.
                                        ;This test is repeated 15 times in order to test each index register.

                    001006              SN=SN+1
                    000006                      ZZ=ZZ+1
031277  201 06 0 00 000006                      MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
031300  505 06 0 00 000006                      HRLI    ZZ,ZZ                   ;Pass if index register contains its address
031301  312 06 0 06 000000                      CAME    ZZ,(ZZ)                 ;In both halves
                                                STOP^
031302  254 04 0 00 031303                      HALT    .+1                     ;Test failed if program halts here
031303  324 00 0 00 031304                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test verifies that all index registers index correctly. first, both
                                        ;Halves of the index register are preloaded with its  address.  then, the
                                        ;Index register is referenced. it is then checked for its address in both
                                        ;Halves.  if it contains its address in both halves this test passes.
                                        ;This test is repeated 15 times in order to test each index register.

                    001007              SN=SN+1
                    000007                      ZZ=ZZ+1
031304  201 07 0 00 000007                      MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
031305  505 07 0 00 000007                      HRLI    ZZ,ZZ                   ;Pass if index register contains its address
031306  312 07 0 07 000000                      CAME    ZZ,(ZZ)                 ;In both halves
                                                STOP^
031307  254 04 0 00 031310                      HALT    .+1                     ;Test failed if program halts here
031310  324 00 0 00 031311                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test verifies that all index registers index correctly. first, both
                                        ;Halves of the index register are preloaded with its  address.  then, the
                                        ;Index register is referenced. it is then checked for its address in both
                                        ;Halves.  if it contains its address in both halves this test passes.
                                        ;This test is repeated 15 times in order to test each index register.

                    001010              SN=SN+1
                    000010                      ZZ=ZZ+1
031311  201 10 0 00 000010                      MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
031312  505 10 0 00 000010                      HRLI    ZZ,ZZ                   ;Pass if index register contains its address
031313  312 10 0 10 000000                      CAME    ZZ,(ZZ)                 ;In both halves
                                                STOP^
031314  254 04 0 00 031315                      HALT    .+1                     ;Test failed if program halts here
031315  324 00 0 00 031316                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test verifies that all index registers index correctly. first, both
```

```
2002                                    ;Halves of the index register are preloaded with its  address.  then, the
2003                                    ;Index register is referenced. it is then checked for its address in both
2004                                    ;Halves.  if it contains its address in both halves this test passes.
2005                                    ;This test is repeated 15 times in order to test each index register.
2006
2007                    001011          SN=SN+1
2008                    000011                  ZZ=ZZ+1
2009  031316  201 11 0 00 000011              MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
2010  031317  505 11 0 00 000011              HRLI    ZZ,ZZ                   ;Pass if index register contains its address
2011  031320  312 11 0 11 000000              CAME    ZZ,(ZZ)                 ;In both halves
2012                                          STOP^
2013  031321  254 04 0 00 031322              HALT    .+1                     ;Test failed if program halts here
2014  031322  324 00 0 00 031323              JUMPA   .+1                     ;To loop change this instruction^
2015
2016                                    ;**********
2017
2018
2019                                    ;This test verifies that all index registers index correctly. first, both
2020                                    ;Halves of the index register are preloaded with its  address.  then, the
2021                                    ;Index register is referenced. it is then checked for its address in both
2022                                    ;Halves.  if it contains its address in both halves this test passes.
2023                                    ;This test is repeated 15 times in order to test each index register.
2024
2025                    001012          SN=SN+1
2026                    000012                  ZZ=ZZ+1
2027  031323  201 12 0 00 000012              MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
2028  031324  505 12 0 00 000012              HRLI    ZZ,ZZ                   ;Pass if index register contains its address
2029  031325  312 12 0 12 000000              CAME    ZZ,(ZZ)                 ;In both halves
2030                                          STOP^
2031  031326  254 04 0 00 031327              HALT    .+1                     ;Test failed if program halts here
2032  031327  324 00 0 00 031330              JUMPA   .+1                     ;To loop change this instruction^
2033
2034                                    ;**********
2035
2036
2037                                    ;This test verifies that all index registers index correctly. first, both
2038                                    ;Halves of the index register are preloaded with its  address.  then, the
2039                                    ;Index register is referenced. it is then checked for its address in both
2040                                    ;Halves.  if it contains its address in both halves this test passes.
2041                                    ;This test is repeated 15 times in order to test each index register.
2042
2043                    001013          SN=SN+1
2044                    000013                  ZZ=ZZ+1
2045  031330  201 13 0 00 000013              MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
2046  031331  505 13 0 00 000013              HRLI    ZZ,ZZ                   ;Pass if index register contains its address
2047  031332  312 13 0 13 000000              CAME    ZZ,(ZZ)                 ;In both halves
2048                                          STOP^
2049  031333  254 04 0 00 031334              HALT    .+1                     ;Test failed if program halts here
2050  031334  324 00 0 00 031335              JUMPA   .+1                     ;To loop change this instruction^
2051
2052                                    ;**********
2053
2054
2055                                    ;This test verifies that all index registers index correctly. first, both
2056                                    ;Halves of the index register are preloaded with its  address.  then, the
```

```
                                ;Index register is referenced. it is then checked for its address in both
                                ;Halves.  if it contains its address in both halves this test passes.
                                ;This test is repeated 15 times in order to test each index register.

                        001014  SN=SN+1
                        000014          ZZ=ZZ+1
031335  201 14 0 00 000014              MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
031336  505 14 0 00 000014              HRLI    ZZ,ZZ                   ;Pass if index register contains its address
031337  312 14 0 14 000000              CAME    ZZ,(ZZ)                 ;In both halves
                                        STOP^
031340  254 04 0 00 031341              HALT    .+1                     ;Test failed if program halts here
031341  324 00 0 00 031342              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********


                                ;This test verifies that all index registers index correctly. first, both
                                ;Halves of the index register are preloaded with its  address.  then, the
                                ;Index register is referenced. it is then checked for its address in both
                                ;Halves.  if it contains its address in both halves this test passes.
                                ;This test is repeated 15 times in order to test each index register.

                        001015  SN=SN+1
                        000015          ZZ=ZZ+1
031342  201 15 0 00 000015              MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
031343  505 15 0 00 000015              HRLI    ZZ,ZZ                   ;Pass if index register contains its address
031344  312 15 0 15 000000              CAME    ZZ,(ZZ)                 ;In both halves
                                        STOP^
031345  254 04 0 00 031346              HALT    .+1                     ;Test failed if program halts here
031346  324 00 0 00 031347              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********


                                ;This test verifies that all index registers index correctly. first, both
                                ;Halves of the index register are preloaded with its  address.  then, the
                                ;Index register is referenced. it is then checked for its address in both
                                ;Halves.  if it contains its address in both halves this test passes.
                                ;This test is repeated 15 times in order to test each index register.

                        001016  SN=SN+1
                        000016          ZZ=ZZ+1
031347  201 16 0 00 000016              MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
031350  505 16 0 00 000016              HRLI    ZZ,ZZ                   ;Pass if index register contains its address
031351  312 16 0 16 000000              CAME    ZZ,(ZZ)                 ;In both halves
                                        STOP^
031352  254 04 0 00 031353              HALT    .+1                     ;Test failed if program halts here
031353  324 00 0 00 031354              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********


                                ;This test verifies that all index registers index correctly. first, both
                                ;Halves of the index register are preloaded with its  address.  then, the
                                ;Index register is referenced. it is then checked for its address in both
```

```
2112                                    ;Halves.  if it contains its address in both halves this test passes.
2113                                    ;This test is repeated 15 times in order to test each index register.
2114
2115                    001017          SN=SN+1
2116                    000017                  ZZ=ZZ+1
2117  031354  201 17 0 00 000017               MOVEI   ZZ,ZZ                   ;Preload both halves of index reg with its address
2118  031355  505 17 0 00 000017               HRLI    ZZ,ZZ                   ;Pass if index register contains its address
2119  031356  312 17 0 17 000000               CAME    ZZ,(ZZ)                 ;In both halves
2120                                            STOP^
2121  031357  254 04 0 00 031360               HALT    .+1                     ;Test failed if program halts here
2122  031360  324 00 0 00 031361               JUMPA   .+1                     ;To loop change this instruction^
2123
2124                                    ;**********
2125
```

```
                                SUBTTL  TEST OF INDEX REGISTER ADDRESSING

                                ;**********

                                ;This  test  verifies that index register addressing functions correctly.
                                ;In this test, a move  instruction  using indexing is executed.  the move
                                ;Instruction should place the contents of the location specified by index
                                ;Register 2 into ac3.  since index  register 2 was preloaded with a 1 and
                                ;Ac1 was preloaded with a 0, the  final  result  in  ac3 should be 0.  if
                                ;C(ac3)=0, this test passes.

031361  476 00 0 00 000003      C1100:  SETOM   3                       ;Preload ac3 with -1,,1
031362  402 00 0 00 000001              SETZM   1                       ;Preload ac1 with 0
031363  201 02 0 00 000001              MOVEI   2,1                     ;Setup index register 2 with 1
031364  200 03 0 02 000000              MOVE    3,(2)                   ;*Fwt from indexed location
031365  332 00 0 00 000003              SKIPE   3                       ;Test indexing
                                        STOP^
031366  254 04 0 00 031367              HALT    .+1                     ;Test failed if program halts here
031367  324 00 0 00 031370              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                        001200  SN=1200
                        000000          ZZ=0

                                C1200:  REPEAT  ^D18,<
                                ;This test  verifies  that index register addressing functions correctly.
                                ;In this test, a move  instruction  using indexing is executed.  the move
                                ;Instruction should place the contents of the location specified by index
                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
                                ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                                SN=SN+1
                                        ZZ=ZZ+ZZ
                                        IFE     ZZ,<ZZ=1>
                                        SETOM   3                       ;Preload ac3 with -1,,-1
                                        MOVEI   1,ZZ                    ;Preload ac1 with floating 1
                                        MOVEI   2,1                     ;Setup index register
                                        MOVE    3,(2)                   ;*Fwt from indexed location
                                        CAIE    3,ZZ                    ;Test indexing
                                        STOP

                                ;**********
                                >

                                ;This test  verifies  that index register addressing functions correctly.
                                ;In this test, a move  instruction  using indexing is executed.  the move
                                ;Instruction should place the contents of the location specified by index
                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
                                ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                        001201  SN=SN+1
                        000000          ZZ=ZZ+ZZ
```

```
2181                        000001              IFE     ZZ,<ZZ=1>
2182  031370  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2183  031371  201 01 0 00 000001                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2184  031372  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2185  031373  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2186  031374  302 03 0 00 000001                CAIE    3,ZZ                    ;Test indexing
2187                                            STOP^
2188  031375  254 04 0 00 031376                HALT    .+1                     ;Test failed if program halts here
2189  031376  324 00 0 00 031377                JUMPA   .+1                     ;To loop change this instruction^
2190
2191                                      ;**********
2192
2193
2194                                      ;This test  verifies  that index register addressing functions correctly.
2195                                      ;In this test, a move  instruction  using indexing is executed.  the move
2196                                      ;Instruction should place the contents of the location specified by index
2197                                      ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2198                                      ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2199                                      ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2200
2201                        001202        SN=SN+1
2202                        000002              ZZ=ZZ+ZZ
2203                                            IFE     ZZ,<ZZ=1>
2204  031377  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2205  031400  201 01 0 00 000002                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2206  031401  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2207  031402  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2208  031403  302 03 0 00 000002                CAIE    3,ZZ                    ;Test indexing
2209                                            STOP^
2210  031404  254 04 0 00 031405                HALT    .+1                     ;Test failed if program halts here
2211  031405  324 00 0 00 031406                JUMPA   .+1                     ;To loop change this instruction^
2212
2213                                      ;**********
2214
2215
2216                                      ;This test  verifies  that index register addressing functions correctly.
2217                                      ;In this test, a move  instruction  using indexing is executed.  the move
2218                                      ;Instruction should place the contents of the location specified by index
2219                                      ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2220                                      ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2221                                      ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2222
2223                        001203        SN=SN+1
2224                        000004              ZZ=ZZ+ZZ
2225                                            IFE     ZZ,<ZZ=1>
2226  031406  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2227  031407  201 01 0 00 000004                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2228  031410  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2229  031411  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2230  031412  302 03 0 00 000004                CAIE    3,ZZ                    ;Test indexing
2231                                            STOP^
2232  031413  254 04 0 00 031414                HALT    .+1                     ;Test failed if program halts here
2233  031414  324 00 0 00 031415                JUMPA   .+1                     ;To loop change this instruction^
2234
2235                                      ;**********
```

```
2236
2237
2238                                    ;This test  verifies  that index register addressing functions correctly.
2239                                    ;In this test, a move  instruction  using indexing is executed.  the move
2240                                    ;Instruction should place the contents of the location specified by index
2241                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2242                                    ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2243                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2244
2245                    001204          SN=SN+1
2246                    000010                  ZZ=ZZ+ZZ
2247                                            IFE     ZZ,<ZZ=1>
2248  031415  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2249  031416  201 01 0 00 000010                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2250  031417  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2251  031420  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2252  031421  302 03 0 00 000010                CAIE    3,ZZ                    ;Test indexing
2253                                            STOP^
2254  031422  254 04 0 00 031423                HALT    .+1                     ;Test failed if program halts here
2255  031423  324 00 0 00 031424                JUMPA   .+1                     ;To loop change this instruction^
2256
2257                                    ;**********
2258
2259
2260                                    ;This test  verifies  that index register addressing functions correctly.
2261                                    ;In this test, a move  instruction  using indexing is executed.  the move
2262                                    ;Instruction should place the contents of the location specified by index
2263                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2264                                    ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2265                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2266
2267                    001205          SN=SN+1
2268                    000020                  ZZ=ZZ+ZZ
2269                                            IFE     ZZ,<ZZ=1>
2270  031424  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2271  031425  201 01 0 00 000020                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2272  031426  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2273  031427  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2274  031430  302 03 0 00 000020                CAIE    3,ZZ                    ;Test indexing
2275                                            STOP^
2276  031431  254 04 0 00 031432                HALT    .+1                     ;Test failed if program halts here
2277  031432  324 00 0 00 031433                JUMPA   .+1                     ;To loop change this instruction^
2278
2279                                    ;**********
2280
2281
2282                                    ;This test  verifies  that index register addressing functions correctly.
2283                                    ;In this test, a move  instruction  using indexing is executed.  the move
2284                                    ;Instruction should place the contents of the location specified by index
2285                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2286                                    ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2287                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2288
2289                    001206          SN=SN+1
2290                    000040                  ZZ=ZZ+ZZ
```

```
2291                                            IFE     ZZ,<ZZ=1>
2292  031433  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2293  031434  201 01 0 00 000040                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2294  031435  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2295  031436  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2296  031437  302 03 0 00 000040                CAIE    3,ZZ                    ;Test indexing
2297                                            STOP^
2298  031440  254 04 0 00 031441                HALT    .+1                     ;Test failed if program halts here
2299  031441  324 00 0 00 031442                JUMPA   .+1                     ;To loop change this instruction^
2300
2301                                    ;**********
2302
2303
2304                                    ;This test  verifies  that index register addressing functions correctly.
2305                                    ;In this test, a move  instruction  using indexing is executed.  the move
2306                                    ;Instruction should place the contents of the location specified by index
2307                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2308                                    ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2309                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2310
2311                     001207         SN=SN+1
2312                     000100                 ZZ=ZZ+ZZ
2313                                            IFE     ZZ,<ZZ=1>
2314  031442  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2315  031443  201 01 0 00 000100                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2316  031444  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2317  031445  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2318  031446  302 03 0 00 000100                CAIE    3,ZZ                    ;Test indexing
2319                                            STOP^
2320  031447  254 04 0 00 031450                HALT    .+1                     ;Test failed if program halts here
2321  031450  324 00 0 00 031451                JUMPA   .+1                     ;To loop change this instruction^
2322
2323                                    ;**********
2324
2325
2326                                    ;This test  verifies  that index register addressing functions correctly.
2327                                    ;In this test, a move  instruction  using indexing is executed.  the move
2328                                    ;Instruction should place the contents of the location specified by index
2329                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2330                                    ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2331                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2332
2333                     001210         SN=SN+1
2334                     000200                 ZZ=ZZ+ZZ
2335                                            IFE     ZZ,<ZZ=1>
2336  031451  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2337  031452  201 01 0 00 000200                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2338  031453  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2339  031454  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2340  031455  302 03 0 00 000200                CAIE    3,ZZ                    ;Test indexing
2341                                            STOP^
2342  031456  254 04 0 00 031457                HALT    .+1                     ;Test failed if program halts here
2343  031457  324 00 0 00 031460                JUMPA   .+1                     ;To loop change this instruction^
2344
2345                                    ;**********
```

```
2346
2347
2348                                    ;This test  verifies  that index register addressing functions correctly.
2349                                    ;In this test, a move  instruction  using indexing is executed.  the move
2350                                    ;Instruction should place the contents of the location specified by index
2351                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2352                                    ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2353                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes
2354
2355                 001211             SN=SN+1
2356                 000400                     ZZ=ZZ+ZZ
2357                                            IFE     ZZ,<ZZ=1>
2358  031460  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2359  031461  201 01 0 00 000400                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2360  031462  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2361  031463  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2362  031464  302 03 0 00 000400                CAIE    3,ZZ                    ;Test indexing
2363                                            STOP^
2364  031465  254 04 0 00 031466                HALT    .+1                     ;Test failed if program halts here
2365  031466  324 00 0 00 031467                JUMPA   .+1                     ;To loop change this instruction^
2366
2367                                    ;**********
2368
2369
2370                                    ;This test  verifies  that index register addressing functions correctly.
2371                                    ;In this test, a move  instruction  using indexing is executed.  the move
2372                                    ;Instruction should place the contents of the location specified by index
2373                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2374                                    ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2375                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2376
2377                 001212             SN=SN+1
2378                 001000                     ZZ=ZZ+ZZ
2379                                            IFE     ZZ,<ZZ=1>
2380  031467  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2381  031470  201 01 0 00 001000                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2382  031471  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2383  031472  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2384  031473  302 03 0 00 001000                CAIE    3,ZZ                    ;Test indexing
2385                                            STOP^
2386  031474  254 04 0 00 031475                HALT    .+1                     ;Test failed if program halts here
2387  031475  324 00 0 00 031476                JUMPA   .+1                     ;To loop change this instruction^
2388
2389                                    ;**********
2390
2391
2392                                    ;This test  verifies  that index register addressing functions correctly.
2393                                    ;In this test, a move  instruction  using indexing is executed.  the move
2394                                    ;Instruction should place the contents of the location specified by index
2395                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2396                                    ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2397                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2398
2399                 001213             SN=SN+1
2400                 002000                     ZZ=ZZ+ZZ
```

```
2401                                            IFE     ZZ,<ZZ=1>
2402  031476  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2403  031477  201 01 0 00 002000                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2404  031500  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2405  031501  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2406  031502  302 03 0 00 002000                CAIE    3,ZZ                    ;Test indexing
2407                                            STOP^
2408  031503  254 04 0 00 031504                HALT    .+1                     ;Test failed if program halts here
2409  031504  324 00 0 00 031505                JUMPA   .+1                     ;To loop change this instruction^
2410
2411                                    ;**********
2412
2413
2414                                    ;This test  verifies  that index register addressing functions correctly.
2415                                    ;In this test, a move  instruction  using indexing is executed.  the move
2416                                    ;Instruction should place the contents of the location specified by index
2417                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2418                                    ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2419                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2420
2421                     001214         SN=SN+1
2422                     004000                 ZZ=ZZ+ZZ
2423                                            IFE     ZZ,<ZZ=1>
2424  031505  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2425  031506  201 01 0 00 004000                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2426  031507  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2427  031510  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2428  031511  302 03 0 00 004000                CAIE    3,ZZ                    ;Test indexing
2429                                            STOP^
2430  031512  254 04 0 00 031513                HALT    .+1                     ;Test failed if program halts here
2431  031513  324 00 0 00 031514                JUMPA   .+1                     ;To loop change this instruction^
2432
2433                                    ;**********
2434
2435
2436                                    ;This test  verifies  that index register addressing functions correctly.
2437                                    ;In this test, a move  instruction  using indexing is executed.  the move
2438                                    ;Instruction should place the contents of the location specified by index
2439                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2440                                    ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
2441                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2442
2443                     001215         SN=SN+1
2444                     010000                 ZZ=ZZ+ZZ
2445                                            IFE     ZZ,<ZZ=1>
2446  031514  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2447  031515  201 01 0 00 010000                MOVEI   1,ZZ                    ;Preload ac1 with floating 1
2448  031516  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2449  031517  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2450  031520  302 03 0 00 010000                CAIE    3,ZZ                    ;Test indexing
2451                                            STOP^
2452  031521  254 04 0 00 031522                HALT    .+1                     ;Test failed if program halts here
2453  031522  324 00 0 00 031523                JUMPA   .+1                     ;To loop change this instruction^
2454
2455                                    ;**********
```

```
                                        ;This test  verifies  that index register addressing functions correctly.
                                        ;In this test, a move  instruction  using indexing is executed.  the move
                                        ;Instruction should place the contents of the location specified by index
                                        ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                        ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
                                        ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                    001216              SN=SN+1
                    020000                      ZZ=ZZ+ZZ
                                                IFE     ZZ,<ZZ=1>
031523  476 00 0 00 000003                      SETOM   3                       ;Preload ac3 with -1,,-1
031524  201 01 0 00 020000                      MOVEI   1,ZZ                    ;Preload ac1 with floating 1
031525  201 02 0 00 000001                      MOVEI   2,1                     ;Setup index register
031526  200 03 0 02 000000                      MOVE    3,(2)                   ;*Fwt from indexed location
031527  302 03 0 00 020000                      CAIE    3,ZZ                    ;Test indexing
                                                STOP^
031530  254 04 0 00 031531                      HALT    .+1                     ;Test failed if program halts here
031531  324 00 0 00 031532                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test  verifies  that index register addressing functions correctly.
                                        ;In this test, a move  instruction  using indexing is executed.  the move
                                        ;Instruction should place the contents of the location specified by index
                                        ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                        ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
                                        ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                    001217              SN=SN+1
                    040000                      ZZ=ZZ+ZZ
                                                IFE     ZZ,<ZZ=1>
031532  476 00 0 00 000003                      SETOM   3                       ;Preload ac3 with -1,,-1
031533  201 01 0 00 040000                      MOVEI   1,ZZ                    ;Preload ac1 with floating 1
031534  201 02 0 00 000001                      MOVEI   2,1                     ;Setup index register
031535  200 03 0 02 000000                      MOVE    3,(2)                   ;*Fwt from indexed location
031536  302 03 0 00 040000                      CAIE    3,ZZ                    ;Test indexing
                                                STOP^
031537  254 04 0 00 031540                      HALT    .+1                     ;Test failed if program halts here
031540  324 00 0 00 031541                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test  verifies  that index register addressing functions correctly.
                                        ;In this test, a move  instruction  using indexing is executed.  the move
                                        ;Instruction should place the contents of the location specified by index
                                        ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                        ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
                                        ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                    001220              SN=SN+1
                    100000                      ZZ=ZZ+ZZ
```

```
                                        IFE     ZZ,<ZZ=1>
031541  476 00 0 00 000003              SETOM   3                       ;Preload ac3 with -1,,-1
031542  201 01 0 00 100000              MOVEI   1,ZZ                    ;Preload ac1 with floating 1
031543  201 02 0 00 000001              MOVEI   2,1                     ;Setup index register
031544  200 03 0 02 000000              MOVE    3,(2)                   ;*Fwt from indexed location
031545  302 03 0 00 100000              CAIE    3,ZZ                    ;Test indexing
                                        STOP^
031546  254 04 0 00 031547              HALT    .+1                     ;Test failed if program halts here
031547  324 00 0 00 031550              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********


                                ;This test  verifies  that index register addressing functions correctly.
                                ;In this test, a move  instruction  using indexing is executed.  the move
                                ;Instruction should place the contents of the location specified by index
                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
                                ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                        001221  SN=SN+1
                        200000          ZZ=ZZ+ZZ
                                        IFE     ZZ,<ZZ=1>
031550  476 00 0 00 000003              SETOM   3                       ;Preload ac3 with -1,,-1
031551  201 01 0 00 200000              MOVEI   1,ZZ                    ;Preload ac1 with floating 1
031552  201 02 0 00 000001              MOVEI   2,1                     ;Setup index register
031553  200 03 0 02 000000              MOVE    3,(2)                   ;*Fwt from indexed location
031554  302 03 0 00 200000              CAIE    3,ZZ                    ;Test indexing
                                        STOP^
031555  254 04 0 00 031556              HALT    .+1                     ;Test failed if program halts here
031556  324 00 0 00 031557              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********


                                ;This test  verifies  that index register addressing functions correctly.
                                ;In this test, a move  instruction  using indexing is executed.  the move
                                ;Instruction should place the contents of the location specified by index
                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                ;Ac1 was preloaded with a floating 1. the final result in ac3 should be a
                                ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                        001222  SN=SN+1
                        400000          ZZ=ZZ+ZZ
                                        IFE     ZZ,<ZZ=1>
031557  476 00 0 00 000003              SETOM   3                       ;Preload ac3 with -1,,-1
031560  201 01 0 00 400000              MOVEI   1,ZZ                    ;Preload ac1 with floating 1
031561  201 02 0 00 000001              MOVEI   2,1                     ;Setup index register
031562  200 03 0 02 000000              MOVE    3,(2)                   ;*Fwt from indexed location
031563  302 03 0 00 400000              CAIE    3,ZZ                    ;Test indexing
                                        STOP^
031564  254 04 0 00 031565              HALT    .+1                     ;Test failed if program halts here
031565  324 00 0 00 031566              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
2566
```

```
                        001300          SN=1300
                        000000                  ZZ=0

                                        C1300:  REPEAT  ^D18,<
                                        ;This test  verifies  that index register addressing functions correctly.
                                        ;In this test, a move  instruction  using indexing is executed.  the move
                                        ;Instruction should place the contents of the location specified by index
                                        ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                        ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
                                        ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                                        SN=SN+1
                                                ZZ=ZZ+ZZ
                                                IFE     ZZ,<ZZ=1>
                                                SETOM   3                       ;Preload ac3 with -1,,-1
                                                MOVSI   1,ZZ                    ;Preload ac1 with floating 1
                                                MOVEI   2,1                     ;Setup index register
                                                MOVE    3,(2)                   ;*Fwt from indexed location
                                                CAME    3,[ZZ,,0]               ;Test indexing
                                                STOP

                                        ;**********
                                        >

                                        ;This test  verifies  that index register addressing functions correctly.
                                        ;In this test, a move  instruction  using indexing is executed.  the move
                                        ;Instruction should place the contents of the location specified by index
                                        ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                        ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
                                        ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                        001301          SN=SN+1
                        000000                  ZZ=ZZ+ZZ
                        000001                  IFE     ZZ,<ZZ=1>
031566  476 00 0 00 000003                      SETOM   3                       ;Preload ac3 with -1,,-1
031567  205 01 0 00 000001                      MOVSI   1,ZZ                    ;Preload ac1 with floating 1
031570  201 02 0 00 000001                      MOVEI   2,1                     ;Setup index register
031571  200 03 0 02 000000                      MOVE    3,(2)                   ;*Fwt from indexed location
031572  312 03 0 00 034421                      CAME    3,[ZZ,,0]               ;Test indexing
                                                STOP^
031573  254 04 0 00 031574                      HALT    .+1                     ;Test failed if program halts here
031574  324 00 0 00 031575                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test  verifies  that index register addressing functions correctly.
                                        ;In this test, a move  instruction  using indexing is executed.  the move
                                        ;Instruction should place the contents of the location specified by index
                                        ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                        ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
                                        ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                        001302          SN=SN+1
                        000002                  ZZ=ZZ+ZZ
```

```
2622                                            IFE     ZZ,<ZZ=1>
2623  031575  476 00 0 00 000003               SETOM   3                       ;Preload ac3 with -1,,-1
2624  031576  205 01 0 00 000002               MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2625  031577  201 02 0 00 000001               MOVEI   2,1                     ;Setup index register
2626  031600  200 03 0 02 000000               MOVE    3,(2)                   ;*Fwt from indexed location
2627  031601  312 03 0 00 034422               CAME    3,[ZZ,,0]               ;Test indexing
2628                                            STOP^
2629  031602  254 04 0 00 031603               HALT    .+1                     ;Test failed if program halts here
2630  031603  324 00 0 00 031604               JUMPA   .+1                     ;To loop change this instruction^
2631
2632                                    ;**********
2633
2634
2635                                    ;This test  verifies  that index register addressing functions correctly.
2636                                    ;In this test, a move  instruction  using indexing is executed.  the move
2637                                    ;Instruction should place the contents of the location specified by index
2638                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2639                                    ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
2640                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2641
2642                     001303         SN=SN+1
2643                     000004                 ZZ=ZZ+ZZ
2644                                            IFE     ZZ,<ZZ=1>
2645  031604  476 00 0 00 000003               SETOM   3                       ;Preload ac3 with -1,,-1
2646  031605  205 01 0 00 000004               MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2647  031606  201 02 0 00 000001               MOVEI   2,1                     ;Setup index register
2648  031607  200 03 0 02 000000               MOVE    3,(2)                   ;*Fwt from indexed location
2649  031610  312 03 0 00 034423               CAME    3,[ZZ,,0]               ;Test indexing
2650                                            STOP^
2651  031611  254 04 0 00 031612               HALT    .+1                     ;Test failed if program halts here
2652  031612  324 00 0 00 031613               JUMPA   .+1                     ;To loop change this instruction^
2653
2654                                    ;**********
2655
2656
2657                                    ;This test  verifies  that index register addressing functions correctly.
2658                                    ;In this test, a move  instruction  using indexing is executed.  the move
2659                                    ;Instruction should place the contents of the location specified by index
2660                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2661                                    ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
2662                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2663
2664                     001304         SN=SN+1
2665                     000010                 ZZ=ZZ+ZZ
2666                                            IFE     ZZ,<ZZ=1>
2667  031613  476 00 0 00 000003               SETOM   3                       ;Preload ac3 with -1,,-1
2668  031614  205 01 0 00 000010               MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2669  031615  201 02 0 00 000001               MOVEI   2,1                     ;Setup index register
2670  031616  200 03 0 02 000000               MOVE    3,(2)                   ;*Fwt from indexed location
2671  031617  312 03 0 00 034424               CAME    3,[ZZ,,0]               ;Test indexing
2672                                            STOP^
2673  031620  254 04 0 00 031621               HALT    .+1                     ;Test failed if program halts here
2674  031621  324 00 0 00 031622               JUMPA   .+1                     ;To loop change this instruction^
2675
2676                                    ;**********
```

```
                                        ;This test  verifies  that index register addressing functions correctly.
                                        ;In this test, a move  instruction  using indexing is executed.  the move
                                        ;Instruction should place the contents of the location specified by index
                                        ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                        ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
                                        ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                    001305              SN=SN+1
                    000020                      ZZ=ZZ+ZZ
                                                IFE     ZZ,<ZZ=1>
031622  476 00 0 00 000003                      SETOM   3                       ;Preload ac3 with -1,,-1
031623  205 01 0 00 000020                      MOVSI   1,ZZ                    ;Preload ac1 with floating 1
031624  201 02 0 00 000001                      MOVEI   2,1                     ;Setup index register
031625  200 03 0 02 000000                      MOVE    3,(2)                   ;*Fwt from indexed location
031626  312 03 0 00 034425                      CAME    3,[ZZ,,0]               ;Test indexing
                                                STOP^
031627  254 04 0 00 031630                      HALT    .+1                     ;Test failed if program halts here
031630  324 00 0 00 031631                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test  verifies  that index register addressing functions correctly.
                                        ;In this test, a move  instruction  using indexing is executed.  the move
                                        ;Instruction should place the contents of the location specified by index
                                        ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                        ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
                                        ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                    001306              SN=SN+1
                    000040                      ZZ=ZZ+ZZ
                                                IFE     ZZ,<ZZ=1>
031631  476 00 0 00 000003                      SETOM   3                       ;Preload ac3 with -1,,-1
031632  205 01 0 00 000040                      MOVSI   1,ZZ                    ;Preload ac1 with floating 1
031633  201 02 0 00 000001                      MOVEI   2,1                     ;Setup index register
031634  200 03 0 02 000000                      MOVE    3,(2)                   ;*Fwt from indexed location
031635  312 03 0 00 034426                      CAME    3,[ZZ,,0]               ;Test indexing
                                                STOP^
031636  254 04 0 00 031637                      HALT    .+1                     ;Test failed if program halts here
031637  324 00 0 00 031640                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********


                                        ;This test  verifies  that index register addressing functions correctly.
                                        ;In this test, a move  instruction  using indexing is executed.  the move
                                        ;Instruction should place the contents of the location specified by index
                                        ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                        ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
                                        ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                    001307              SN=SN+1
                    000100                      ZZ=ZZ+ZZ
```

```
                                        IFE     ZZ,<ZZ=1>
031640  476 00 0 00 000003              SETOM   3                       ;Preload ac3 with -1,,-1
031641  205 01 0 00 000100              MOVSI   1,ZZ                    ;Preload ac1 with floating 1
031642  201 02 0 00 000001              MOVEI   2,1                     ;Setup index register
031643  200 03 0 02 000000              MOVE    3,(2)                   ;*Fwt from indexed location
031644  312 03 0 00 034427              CAME    3,[ZZ,,0]               ;Test indexing
                                        STOP^
031645  254 04 0 00 031646              HALT    .+1                     ;Test failed if program halts here
031646  324 00 0 00 031647              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********


                                ;This test  verifies  that index register addressing functions correctly.
                                ;In this test, a move  instruction  using indexing is executed.  the move
                                ;Instruction should place the contents of the location specified by index
                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
                                ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                        001310  SN=SN+1
                        000200          ZZ=ZZ+ZZ
                                        IFE     ZZ,<ZZ=1>
031647  476 00 0 00 000003              SETOM   3                       ;Preload ac3 with -1,,-1
031650  205 01 0 00 000200              MOVSI   1,ZZ                    ;Preload ac1 with floating 1
031651  201 02 0 00 000001              MOVEI   2,1                     ;Setup index register
031652  200 03 0 02 000000              MOVE    3,(2)                   ;*Fwt from indexed location
031653  312 03 0 00 034430              CAME    3,[ZZ,,0]               ;Test indexing
                                        STOP^
031654  254 04 0 00 031655              HALT    .+1                     ;Test failed if program halts here
031655  324 00 0 00 031656              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********


                                ;This test  verifies  that index register addressing functions correctly.
                                ;In this test, a move  instruction  using indexing is executed.  the move
                                ;Instruction should place the contents of the location specified by index
                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
                                ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
                                ;Floating 1.  if c(ac3)=a floating 1, this test passes.

                        001311  SN=SN+1
                        000400          ZZ=ZZ+ZZ
                                        IFE     ZZ,<ZZ=1>
031656  476 00 0 00 000003              SETOM   3                       ;Preload ac3 with -1,,-1
031657  205 01 0 00 000400              MOVSI   1,ZZ                    ;Preload ac1 with floating 1
031660  201 02 0 00 000001              MOVEI   2,1                     ;Setup index register
031661  200 03 0 02 000000              MOVE    3,(2)                   ;*Fwt from indexed location
031662  312 03 0 00 034431              CAME    3,[ZZ,,0]               ;Test indexing
                                        STOP^
031663  254 04 0 00 031664              HALT    .+1                     ;Test failed if program halts here
031664  324 00 0 00 031665              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
2787
2788
2789                                    ;This test  verifies  that index register addressing functions correctly.
2790                                    ;In this test, a move  instruction  using indexing is executed.  the move
2791                                    ;Instruction should place the contents of the location specified by index
2792                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2793                                    ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
2794                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2795
2796                    001312          SN=SN+1
2797                    001000                  ZZ=ZZ+ZZ
2798                                            IFE     ZZ,<ZZ=1>
2799  031665  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2800  031666  205 01 0 00 001000                MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2801  031667  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2802  031670  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2803  031671  312 03 0 00 034432                CAME    3,[ZZ,,0]               ;Test indexing
2804                                            STOP^
2805  031672  254 04 0 00 031673                HALT    .+1                     ;Test failed if program halts here
2806  031673  324 00 0 00 031674                JUMPA   .+1                     ;To loop change this instruction^
2807
2808                                    ;**********
2809
2810
2811                                    ;This test  verifies  that index register addressing functions correctly.
2812                                    ;In this test, a move  instruction  using indexing is executed.  the move
2813                                    ;Instruction should place the contents of the location specified by index
2814                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2815                                    ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
2816                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2817
2818                    001313          SN=SN+1
2819                    002000                  ZZ=ZZ+ZZ
2820                                            IFE     ZZ,<ZZ=1>
2821  031674  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2822  031675  205 01 0 00 002000                MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2823  031676  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2824  031677  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2825  031700  312 03 0 00 034433                CAME    3,[ZZ,,0]               ;Test indexing
2826                                            STOP^
2827  031701  254 04 0 00 031702                HALT    .+1                     ;Test failed if program halts here
2828  031702  324 00 0 00 031703                JUMPA   .+1                     ;To loop change this instruction^
2829
2830                                    ;**********
2831
2832
2833                                    ;This test  verifies  that index register addressing functions correctly.
2834                                    ;In this test, a move  instruction  using indexing is executed.  the move
2835                                    ;Instruction should place the contents of the location specified by index
2836                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2837                                    ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
2838                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2839
2840                    001314          SN=SN+1
2841                    004000                  ZZ=ZZ+ZZ
```

```
2842                                            IFE     ZZ,<ZZ=1>
2843  031703  476 00 0 00 000003               SETOM   3                       ;Preload ac3 with -1,,-1
2844  031704  205 01 0 00 004000               MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2845  031705  201 02 0 00 000001               MOVEI   2,1                     ;Setup index register
2846  031706  200 03 0 02 000000               MOVE    3,(2)                   ;*Fwt from indexed location
2847  031707  312 03 0 00 034434               CAME    3,[ZZ,,0]               ;Test indexing
2848                                            STOP^
2849  031710  254 04 0 00 031711               HALT    .+1                     ;Test failed if program halts here
2850  031711  324 00 0 00 031712               JUMPA   .+1                     ;To loop change this instruction^
2851
2852                                    ;**********
2853
2854
2855                                    ;This test  verifies  that index register addressing functions correctly.
2856                                    ;In this test, a move  instruction  using indexing is executed.  the move
2857                                    ;Instruction should place the contents of the location specified by index
2858                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2859                                    ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
2860                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2861
2862                    001315          SN=SN+1
2863                    010000                  ZZ=ZZ+ZZ
2864                                            IFE     ZZ,<ZZ=1>
2865  031712  476 00 0 00 000003               SETOM   3                       ;Preload ac3 with -1,,-1
2866  031713  205 01 0 00 010000               MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2867  031714  201 02 0 00 000001               MOVEI   2,1                     ;Setup index register
2868  031715  200 03 0 02 000000               MOVE    3,(2)                   ;*Fwt from indexed location
2869  031716  312 03 0 00 034435               CAME    3,[ZZ,,0]               ;Test indexing
2870                                            STOP^
2871  031717  254 04 0 00 031720               HALT    .+1                     ;Test failed if program halts here
2872  031720  324 00 0 00 031721               JUMPA   .+1                     ;To loop change this instruction^
2873
2874                                    ;**********
2875
2876
2877                                    ;This test  verifies  that index register addressing functions correctly.
2878                                    ;In this test, a move  instruction  using indexing is executed.  the move
2879                                    ;Instruction should place the contents of the location specified by index
2880                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2881                                    ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
2882                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2883
2884                    001316          SN=SN+1
2885                    020000                  ZZ=ZZ+ZZ
2886                                            IFE     ZZ,<ZZ=1>
2887  031721  476 00 0 00 000003               SETOM   3                       ;Preload ac3 with -1,,-1
2888  031722  205 01 0 00 020000               MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2889  031723  201 02 0 00 000001               MOVEI   2,1                     ;Setup index register
2890  031724  200 03 0 02 000000               MOVE    3,(2)                   ;*Fwt from indexed location
2891  031725  312 03 0 00 034436               CAME    3,[ZZ,,0]               ;Test indexing
2892                                            STOP^
2893  031726  254 04 0 00 031727               HALT    .+1                     ;Test failed if program halts here
2894  031727  324 00 0 00 031730               JUMPA   .+1                     ;To loop change this instruction^
2895
2896                                    ;**********
```

```
2897
2898
2899                                    ;This test  verifies  that index register addressing functions correctly.
2900                                    ;In this test, a move  instruction  using indexing is executed.  the move
2901                                    ;Instruction should place the contents of the location specified by index
2902                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2903                                    ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
2904                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2905
2906                    001317          SN=SN+1
2907                    040000                  ZZ=ZZ+ZZ
2908                                            IFE     ZZ,<ZZ=1>
2909  031730  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2910  031731  205 01 0 00 040000                MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2911  031732  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2912  031733  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2913  031734  312 03 0 00 034437                CAME    3,[ZZ,,0]               ;Test indexing
2914                                            STOP^
2915  031735  254 04 0 00 031736                HALT    .+1                     ;Test failed if program halts here
2916  031736  324 00 0 00 031737                JUMPA   .+1                     ;To loop change this instruction^
2917
2918                                    ;**********
2919
2920
2921                                    ;This test  verifies  that index register addressing functions correctly.
2922                                    ;In this test, a move  instruction  using indexing is executed.  the move
2923                                    ;Instruction should place the contents of the location specified by index
2924                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2925                                    ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
2926                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2927
2928                    001320          SN=SN+1
2929                    100000                  ZZ=ZZ+ZZ
2930                                            IFE     ZZ,<ZZ=1>
2931  031737  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2932  031740  205 01 0 00 100000                MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2933  031741  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2934  031742  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2935  031743  312 03 0 00 034440                CAME    3,[ZZ,,0]               ;Test indexing
2936                                            STOP^
2937  031744  254 04 0 00 031745                HALT    .+1                     ;Test failed if program halts here
2938  031745  324 00 0 00 031746                JUMPA   .+1                     ;To loop change this instruction^
2939
2940                                    ;**********
2941
2942
2943                                    ;This test  verifies  that index register addressing functions correctly.
2944                                    ;In this test, a move  instruction  using indexing is executed.  the move
2945                                    ;Instruction should place the contents of the location specified by index
2946                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2947                                    ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
2948                                    ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2949
2950                    001321          SN=SN+1
2951                    200000                  ZZ=ZZ+ZZ
```

```
2952                                            IFE     ZZ,<ZZ=1>
2953  031746  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2954  031747  205 01 0 00 200000                MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2955  031750  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2956  031751  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2957  031752  312 03 0 00 034441                CAME    3,[ZZ,,0]               ;Test indexing
2958                                            STOP^
2959  031753  254 04 0 00 031754                HALT    .+1                     ;Test failed if program halts here
2960  031754  324 00 0 00 031755                JUMPA   .+1                     ;To loop change this instruction^
2961
2962                                  ;**********
2963
2964
2965                                  ;This test  verifies  that index register addressing functions correctly.
2966                                  ;In this test, a move  instruction  using indexing is executed.  the move
2967                                  ;Instruction should place the contents of the location specified by index
2968                                  ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2969                                  ;Ac1 was preloaded with a floating 1, the final result in ac3 should be a
2970                                  ;Floating 1.  if c(ac3)=a floating 1, this test passes.
2971
2972                     001322       SN=SN+1
2973                     400000               ZZ=ZZ+ZZ
2974                                          IFE     ZZ,<ZZ=1>
2975  031755  476 00 0 00 000003                SETOM   3                       ;Preload ac3 with -1,,-1
2976  031756  205 01 0 00 400000                MOVSI   1,ZZ                    ;Preload ac1 with floating 1
2977  031757  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2978  031760  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2979  031761  312 03 0 00 034442                CAME    3,[ZZ,,0]               ;Test indexing
2980                                            STOP^
2981  031762  254 04 0 00 031763                HALT    .+1                     ;Test failed if program halts here
2982  031763  32[illegible] 00 0 00 031764      JUMPA   .+1                     ;To loop change this instruction^
2983
2984                                  ;**********
2985
```

```
2986                                    ;This  test  verifies that index register addressing functions correctly.
2987                                    ;In this test, a move instruction  using  indexing is executed.  the move
2988                                    ;Instruction should place the contents of the location specified by index
2989                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
2990                                    ;Ac1 was preloaded with -1,,-1, the final result in ac3 should be -1,,-1.
2991                                    ;If c(ac3)=-1,,-1, this test passes.
2992
2993  031764  402 00 0 00 000003        C1400:  SETZM   3                       ;Preload ac3 with 0
2994  031765  476 00 0 00 000001                SETOM   1                       ;Preload ac1 with -1,,-1
2995  031766  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
2996  031767  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
2997  031770  312 03 0 00 034443                CAME    3,[-1,,-1]              ;Test indexing
2998                                            STOP^
2999  031771  254 04 0 00 031772                HALT    .+1                     ;Test failed if program halts here
3000  031772  324 00 0 00 031773                JUMPA   .+1                     ;To loop change this instruction^
3001
3002                                    ;**********
3003
3004                    001500          SN=1500
3005                    000000                  ZZ=0
3006
3007                                    C1500:  REPEAT  ^D18,<
3008                                    ;This test  verifies  that index register addressing functions correctly.
3009                                    ;In this test, a move instruction  using  indexing is executed.  the move
3010                                    ;Instruction should place the contents of the location specified by index
3011                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3012                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3013                                    ;Floating 0. if c(ac3)=a floating 0, this test passes.
3014
3015                                    SN=SN+1
3016                                            ZZ=<ZZ+ZZ+1>&777777
3017                                            IFE     <ZZ-1>,<ZZ=777776>
3018                                            SETZM   3                       ;Preload ac3 with 0
3019                                            HRROI   1,ZZ                    ;Preload ac1 with floating 0
3020                                            MOVEI   2,1                     ;Setup index register
3021                                            MOVE    3,(2)                   ;*Fwt from indexed location
3022                                            CAME    3,[-1,,ZZ]              ;Test indexing
3023                                            STOP
3024
3025                                    ;**********
3026                                    >
3027
3028                                    ;This test  verifies  that index register addressing functions correctly.
3029                                    ;In this test, a move instruction  using  indexing is executed.  the move
3030                                    ;Instruction should place the contents of the location specified by index
3031                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3032                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3033                                    ;Floating 0. if c(ac3)=a floating 0, this test passes.
3034
3035                    001501          SN=SN+1
3036                    000001                  ZZ=<ZZ+ZZ+1>&777777
3037                    777776                  IFE     <ZZ-1>,<ZZ=777776>
3038  031773  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3039  031774  561 01 0 00 777776                HRROI   1,ZZ                    ;Preload ac1 with floating 0
3040  031775  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
```

```
3041  031776  200 03 0 02 000000            MOVE    3,(2)                   ;*Fwt from indexed location
3042  031777  312 03 0 00 034444            CAME    3,[-1,,ZZ]              ;Test indexing
3043                                        STOP^
3044  032000  254 04 0 00 032001            HALT    .+1                     ;Test failed if program halts here
3045  032001  324 00 0 00 032002            JUMPA   .+1                     ;To loop change this instruction^
3046
3047                                ;**********
3048
3049
3050                                ;This test  verifies  that index register addressing functions correctly.
3051                                ;In this test, a move instruction  using  indexing is executed.  the move
3052                                ;Instruction should place the contents of the location specified by index
3053                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3054                                ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3055                                ;Floating 0. if c(ac3)=a floating 0, this test passes.
3056
3057                    001502      SN=SN+1
3058                    777775              ZZ=<ZZ+ZZ+1>&777777
3059                                        IFE     <ZZ-1>,<ZZ=777776>
3060  032002  402 00 0 00 000003            SETZM   3                       ;Preload ac3 with 0
3061  032003  561 01 0 00 777775            HRROI   1,ZZ                    ;Preload ac1 with floating 0
3062  032004  201 02 0 00 000001            MOVEI   2,1                     ;Setup index register
3063  032005  200 03 0 02 000000            MOVE    3,(2)                   ;*Fwt from indexed location
3064  032006  312 03 0 00 034445            CAME    3,[-1,,ZZ]              ;Test indexing
3065                                        STOP^
3066  032007  254 04 0 00 032010            HALT    .+1                     ;Test failed if program halts here
3067  032010  324 00 0 00 032011            JUMPA   .+1                     ;To loop change this instruction^
3068
3069                                ;**********
3070
3071
3072                                ;This test  verifies  that index register addressing functions correctly.
3073                                ;In this test, a move instruction  using  indexing is executed.  the move
3074                                ;Instruction should place the contents of the location specified by index
3075                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3076                                ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3077                                ;Floating 0. if c(ac3)=a floating 0, this test passes.
3078
3079                    001503      SN=SN+1
3080                    777773              ZZ=<ZZ+ZZ+1>&777777
3081                                        IFE     <ZZ-1>,<ZZ=777776>
3082  032011  402 00 0 00 000003            SETZM   3                       ;Preload ac3 with 0
3083  032012  561 01 0 00 777773            HRROI   1,ZZ                    ;Preload ac1 with floating 0
3084  032013  201 02 0 00 000001            MOVEI   2,1                     ;Setup index register
3085  032014  200 03 0 02 000000            MOVE    3,(2)                   ;*Fwt from indexed location
3086  032015  312 03 0 00 034446            CAME    3,[-1,,ZZ]              ;Test indexing
3087                                        STOP^
3088  032016  254 04 0 00 032017            HALT    .+1                     ;Test failed if program halts here
3089  032017  324 00 0 00 032020            JUMPA   .+1                     ;To loop change this instruction^
3090
3091                                ;**********
3092
3093
3094                                ;This test  verifies  that index register addressing functions correctly.
3095                                ;In this test, a move instruction  using  indexing is executed.  the move
```

```
3096                                    ;Instruction should place the contents of the location specified by index
3097                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3098                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3099                                    ;Floating 0. if c(ac3)=a floating 0, this test passes.
3100
3101                    001504          SN=SN+1
3102                    777767                  ZZ=<ZZ+ZZ+1>&777777
3103                                            IFE     <ZZ-1>,<ZZ=777776>
3104  032020  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3105  032021  561 01 0 00 777767                HRROI   1,ZZ                    ;Preload ac1 with floating 0
3106  032022  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3107  032023  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3108  032024  312 03 0 00 034447                CAME    3,[-1,,ZZ]              ;Test indexing
3109                                            STOP^
3110  032025  254 04 0 00 032026                HALT    .+1                     ;Test failed if program halts here
3111  032026  324 00 0 00 032027                JUMPA   .+1                     ;To loop change this instruction^
3112
3113                                    ;**********
3114
3115
3116                                    ;This test  verifies  that index register addressing functions correctly.
3117                                    ;In this test, a move instruction  using  indexing is executed.  the move
3118                                    ;Instruction should place the contents of the location specified by index
3119                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3120                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3121                                    ;Floating 0. if c(ac3)=a floating 0, this test passes.
3122
3123                    001505          SN=SN+1
3124                    777757                  ZZ=<ZZ+ZZ+1>&777777
3125                                            IFE     <ZZ-1>,<ZZ=777776>
3126  032027  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3127  032030  561 01 0 00 777757                HRROI   1,ZZ                    ;Preload ac1 with floating 0
3128  032031  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3129  032032  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3130  032033  312 03 0 00 034450                CAME    3,[-1,,ZZ]              ;Test indexing
3131                                            STOP^
3132  032034  254 04 0 00 032035                HALT    .+1                     ;Test failed if program halts here
3133  032035  324 00 0 00 032036                JUMPA   .+1                     ;To loop change this instruction^
3134
3135                                    ;**********
3136
3137
3138                                    ;This test  verifies  that index register addressing functions correctly.
3139                                    ;In this test, a move instruction  using  indexing is executed.  the move
3140                                    ;Instruction should place the contents of the location specified by index
3141                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3142                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3143                                    ;Floating 0. if c(ac3)=a floating 0, this test passes.
3144
3145                    001506          SN=SN+1
3146                    777737                  ZZ=<ZZ+ZZ+1>&777777
3147                                            IFE     <ZZ-1>,<ZZ=777776>
3148  032036  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3149  032037  561 01 0 00 777737                HRROI   1,ZZ                    ;Preload ac1 with floating 0
3150  032040  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
```

```
3151  032041  200 03 0 02 000000          MOVE   3,(2)                    ;*Fwt from indexed location
3152  032042  312 03 0 00 034451          CAME   3,[-1,,ZZ]               ;Test indexing
3153                                      STOP^
3154  032043  254 04 0 00 032044          HALT   .+1                      ;Test failed if program halts here
3155  032044  324 00 0 00 032045          JUMPA  .+1                      ;To loop change this instruction^
3156
3157                                ;**********
3158
3159
3160                                ;This test  verifies  that index register addressing functions correctly.
3161                                ;In this test, a move instruction  using  indexing is executed.  the move
3162                                ;Instruction should place the contents of the location specified by index
3163                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3164                                ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3165                                ;Floating 0. if c(ac3)=a floating 0, this test passes.
3166
3167                    001507      SN=SN+1
3168                    777677              ZZ=<ZZ+ZZ+1>&777777
3169                                        IFE    <ZZ-1>,<ZZ=777776>
3170  032045  402 00 0 00 000003            SETZM  3                       ;Preload ac3 with 0
3171  032046  561 01 0 00 777677            HRROI  1,ZZ                    ;Preload ac1 with floating 0
3172  032047  201 02 0 00 000001            MOVEI  2,1                     ;Setup index register
3173  032050  200 03 0 02 000000            MOVE   3,(2)                   ;*Fwt from indexed location
3174  032051  312 03 0 00 034452            CAME   3,[-1,,ZZ]              ;Test indexing
3175                                        STOP^
3176  032052  254 04 0 00 032053            HALT   .+1                     ;Test failed if program halts here
3177  032053  324 00 0 00 032054            JUMPA  .+1                     ;To loop change this instruction^
3178
3179                                ;**********
3180
3181
3182                                ;This test  verifies  that index register addressing functions correctly.
3183                                ;In this test, a move instruction  using  indexing is executed.  the move
3184                                ;Instruction should place the contents of the location specified by index
3185                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3186                                ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3187                                ;Floating 0. if c(ac3)=a floating 0, this test passes.
3188
3189                    001510      SN=SN+1
3190                    777577              ZZ=<ZZ+ZZ+1>&777777
3191                                        IFE    <ZZ-1>,<ZZ=777776>
3192  032054  402 00 0 00 000003            SETZM  3                       ;Preload ac3 with 0
3193  032055  561 01 0 00 777577            HRROI  1,ZZ                    ;Preload ac1 with floating 0
3194  032056  201 02 0 00 000001            MOVEI  2,1                     ;Setup index register
3195  032057  200 03 0 02 000000            MOVE   3,(2)                   ;*Fwt from indexed location
3196  032060  312 03 0 00 034453            CAME   3,[-1,,ZZ]              ;Test indexing
3197                                        STOP^
3198  032061  254 04 0 00 032062            HALT   .+1                     ;Test failed if program halts here
3199  032062  324 00 0 00 032063            JUMPA  .+1                     ;To loop change this instruction^
3200
3201                                ;**********
3202
3203
3204                                ;This test  verifies  that index register addressing functions correctly.
3205                                ;In this test, a move instruction  using  indexing is executed.  the move
```

```
3206                                    ;Instruction should place the contents of the location specified by index
3207                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3208                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3209                                    ;Floating 0. if c(ac3)=a floating 0, this test passes.
3210
3211                    001511          SN=SN+1
3212                    777377                  ZZ=<ZZ+ZZ+1>&777777
3213                                            IFE     <ZZ-1>,<ZZ=777776>
3214  032063  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3215  032064  561 01 0 00 777377                HRROI   1,ZZ                    ;Preload ac1 with floating 0
3216  032065  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3217  032066  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3218  032067  312 03 0 00 034454                CAME    3,[-1,,ZZ]              ;Test indexing
3219                                            STOP^
3220  032070  254 04 0 00 032071                HALT    .+1                     ;Test failed if program halts here
3221  032071  324 00 0 00 032072                JUMPA   .+1                     ;To loop change this instruction^
3222
3223                                    ;**********
3224
3225
3226                                    ;This test  verifies  that index register addressing functions correctly.
3227                                    ;In this test, a move instruction  using  indexing is executed.  the move
3228                                    ;Instruction should place the contents of the location specified by index
3229                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3230                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3231                                    ;Floating 0. if c(ac3)=a floating 0, this test passes.
3232
3233                    001512          SN=SN+1
3234                    776777                  ZZ=<ZZ+ZZ+1>&777777
3235                                            IFE     <ZZ-1>,<ZZ=777776>
3236  032072  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3237  032073  561 01 0 00 776777                HRROI   1,ZZ                    ;Preload ac1 with floating 0
3238  032074  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3239  032075  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3240  032076  312 03 0 00 034455                CAME    3,[-1,,ZZ]              ;Test indexing
3241                                            STOP^
3242  032077  254 04 0 00 032100                HALT    .+1                     ;Test failed if program halts here
3243  032100  324 00 0 00 032101                JUMPA   .+1                     ;To loop change this instruction^
3244
3245                                    ;**********
3246
3247
3248                                    ;This test  verifies  that index register addressing functions correctly.
3249                                    ;In this test, a move instruction  using  indexing is executed.  the move
3250                                    ;Instruction should place the contents of the location specified by index
3251                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3252                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3253                                    ;Floating 0. if c(ac3)=a floating 0, this test passes.
3254
3255                    001513          SN=SN+1
3256                    775777                  ZZ=<ZZ+ZZ+1>&777777
3257                                            IFE     <ZZ-1>,<ZZ=777776>
3258  032101  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3259  032102  561 01 0 00 775777                HRROI   1,ZZ                    ;Preload ac1 with floating 0
3260  032103  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
```

```
3261  032104  200 03 0 02 000000          MOVE    3,(2)                   ;*Fwt from indexed location
3262  032105  312 03 0 00 034456          CAME    3,[-1,,ZZ]              ;Test indexing
3263                                      STOP^
3264  032106  254 04 0 00 032107          HALT    .+1                     ;Test failed if program halts here
3265  032107  324 00 0 00 032110          JUMPA   .+1                     ;To loop change this instruction^
3266
3267                                ;**********
3268
3269
3270                                ;This test  verifies  that index register addressing functions correctly.
3271                                ;In this test, a move instruction  using  indexing is executed.  the move
3272                                ;Instruction should place the contents of the location specified by index
3273                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3274                                ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3275                                ;Floating 0. if c(ac3)=a floating 0, this test passes.
3276
3277                    001514      SN=SN+1
3278                    773777              ZZ=<ZZ+ZZ+1>&777777
3279                                        IFE     <ZZ-1>,<ZZ=777776>
3280  032110  402 00 0 00 000003            SETZM   3                       ;Preload ac3 with 0
3281  032111  561 01 0 00 773777            HRROI   1,ZZ                    ;Preload ac1 with floating 0
3282  032112  201 02 0 00 000001            MOVEI   2,1                     ;Setup index register
3283  032113  200 03 0 02 000000            MOVE    3,(2)                   ;*Fwt from indexed location
3284  032114  312 03 0 00 034457            CAME    3,[-1,,ZZ]              ;Test indexing
3285                                        STOP^
3286  032115  254 04 0 00 032116            HALT    .+1                     ;Test failed if program halts here
3287  032116  324 00 0 00 032117            JUMPA   .+1                     ;To loop change this instruction^
3288
3289                                ;**********
3290
3291
3292                                ;This test  verifies  that index register addressing functions correctly.
3293                                ;In this test, a move instruction  using  indexing is executed.  the move
3294                                ;Instruction should place the contents of the location specified by index
3295                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3296                                ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3297                                ;Floating 0. if c(ac3)=a floating 0, this test passes.
3298
3299                    001515      SN=SN+1
3300                    767777              ZZ=<ZZ+ZZ+1>&777777
3301                                        IFE     <ZZ-1>,<ZZ=777776>
3302  032117  402 00 0 00 000003            SETZM   3                       ;Preload ac3 with 0
3303  032120  561 01 0 00 767777            HRROI   1,ZZ                    ;Preload ac1 with floating 0
3304  032121  201 02 0 00 000001            MOVEI   2,1                     ;Setup index register
3305  032122  200 03 0 02 000000            MOVE    3,(2)                   ;*Fwt from indexed location
3306  032123  312 03 0 00 034460            CAME    3,[-1,,ZZ]              ;Test indexing
3307                                        STOP^
3308  032124  254 04 0 00 032125            HALT    .+1                     ;Test failed if program halts here
3309  032125  324 00 0 00 032126            JUMPA   .+1                     ;To loop change this instruction^
3310
3311                                ;**********
3312
3313
3314                                ;This test  verifies  that index register addressing functions correctly.
3315                                ;In this test, a move instruction  using  indexing is executed.  the move
```

```
3316                                    ;Instruction should place the contents of the location specified by index
3317                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3318                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3319                                    ;Floating 0. if c(ac3)=a floating 0, this test passes.
3320
3321                    001516          SN=SN+1
3322                    757777                  ZZ=<ZZ+ZZ+1>&777777
3323                                            IFE     <ZZ-1>,<ZZ=777776>
3324  032126  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3325  032127  561 01 0 00 757777                HRROI   1,ZZ                    ;Preload ac1 with floating 0
3326  032130  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3327  032131  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3328  032132  312 03 0 00 034461                CAME    3,[-1,,ZZ]              ;Test indexing
3329                                            STOP^
3330  032133  254 04 0 00 032134                HALT    .+1                     ;Test failed if program halts here
3331  032134  324 00 0 00 032135                JUMPA   .+1                     ;To loop change this instruction^
3332
3333                                    ;*********
3334
3335
3336                                    ;This test  verifies  that index register addressing functions correctly.
3337                                    ;In this test, a move instruction  using  indexing is executed.  the move
3338                                    ;Instruction should place the contents of the location specified by index
3339                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3340                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3341                                    ;Floating 0. if c(ac3)=a floating 0, this test passes.
3342
3343                    001517          SN=SN+1
3344                    737777                  ZZ=<ZZ+ZZ+1>&777777
3345                                            IFE     <ZZ-1>,<ZZ=777776>
3346  032135  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3347  032136  561 01 0 00 737777                HRROI   1,ZZ                    ;Preload ac1 with floating 0
3348  032137  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3349  032140  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3350  032141  312 03 0 00 034462                CAME    3,[-1,,ZZ]              ;Test indexing
3351                                            STOP^
3352  032142  254 04 0 00 032143                HALT    .+1                     ;Test failed if program halts here
3353  032143  324 00 0 00 032144                JUMPA   .+1                     ;To loop change this instruction^
3354
3355                                    ;**********
3356
3357
3358                                    ;This test  verifies  that index register addressing functions correctly.
3359                                    ;In this test, a move instruction  using  indexing is executed.  the move
3360                                    ;Instruction should place the contents of the location specified by index
3361                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3362                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3363                                    ;Floating 0. if c(ac3)=a floating 0, this test passes.
3364
3365                    001520          SN=SN+1
3366                    677777                  ZZ=<ZZ+ZZ+1>&777777
3367                                            IFE     <ZZ-1>,<ZZ=777776>
3368  032144  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3369  032145  561 01 0 00 677777                HRROI   1,ZZ                    ;Preload ac1 with floating 0
3370  032146  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
```

```
3371  032147  200 03 0 02 000000          MOVE    3,(2)                   ;*Fwt from indexed location
3372  032150  312 03 0 00 034463          CAME    3,[-1,,ZZ]              ;Test indexing
3373                                      STOP^
3374  032151  254 04 0 00 032152          HALT    .+1                     ;Test failed if program halts here
3375  032152  324 00 0 00 032153          JUMPA   .+1                     ;To loop change this instruction^
3376
3377                                ;**********
3378
3379
3380                                ;This test  verifies  that index register addressing functions correctly.
3381                                ;In this test, a move instruction  using  indexing is executed.  the move
3382                                ;Instruction should place the contents of the location specified by index
3383                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3384                                ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3385                                ;Floating 0. if c(ac3)=a floating 0, this test passes.
3386
3387                    001521      SN=SN+1
3388                    577777              ZZ=<ZZ+ZZ+1>&777777
3389                                        IFE     <ZZ-1>,<ZZ=777776>
3390  032153  402 00 0 00 000003            SETZM   3                       ;Preload ac3 with 0
3391  032154  561 01 0 00 577777            HRROI   1,ZZ                    ;Preload ac1 with floating 0
3392  032155  201 02 0 00 000001            MOVEI   2,1                     ;Setup index register
3393  032156  200 03 0 02 000000            MOVE    3,(2)                   ;*Fwt from indexed location
3394  032157  312 03 0 00 034464            CAME    3,[-1,,ZZ]              ;Test indexing
3395                                        STOP^
3396  032160  254 04 0 00 032161            HALT    .+1                     ;Test failed if program halts here
3397  032161  324 00 0 00 032162            JUMPA   .+1                     ;To loop change this instruction^
3398
3399                                ;**********
3400
3401
3402                                ;This test  verifies  that index register addressing functions correctly.
3403                                ;In this test, a move instruction  using  indexing is executed.  the move
3404                                ;Instruction should place the contents of the location specified by index
3405                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3406                                ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3407                                ;Floating 0. if c(ac3)=a floating 0, this test passes.
3408
3409                    001522      SN=SN+1
3410                    377777              ZZ=<ZZ+ZZ+1>&777777
3411                                        IFE     <ZZ-1>,<ZZ=777776>
3412  032162  402 00 0 00 000003            SETZM   3                       ;Preload ac3 with 0
3413  032163  561 01 0 00 377777            HRROI   1,ZZ                    ;Preload ac1 with floating 0
3414  032164  201 02 0 00 000001            MOVEI   2,1                     ;Setup index register
3415  032165  200 03 0 02 000000            MOVE    3,(2)                   ;*Fwt from indexed location
3416  032166  312 03 0 00 034465            CAME    3,[-1,,ZZ]              ;Test indexing
3417                                        STOP^
3418  032167  254 04 0 00 032170            HALT    .+1                     ;Test failed if program halts here
3419  032170  324 00 0 00 032171            JUMPA   .+1                     ;To loop change this instruction^
3420
3421                                ;**********
3422
```

```
3423                    001600          SN=1600
3424                    000000                  ZZ=0
3425
3426                                    C1600:  REPEAT  ^D18,<
3427                                    ;This  test  verifies that index register addressing functions correctly.
3428                                    ;In this test,  a move  instruction using indexing is executed.  the move
3429                                    ;Instruction should place the contents of the location specified by index
3430                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3431                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3432                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3433
3434                                    SN=SN+1
3435                                            ZZ=<ZZ+ZZ+1>&777777
3436                                            IFE     <ZZ-1>,<ZZ=777776>
3437                                            SETZM   3                       ;Preload ac3 with 0
3438                                            HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3439                                            MOVEI   2,1                     ;Setup index register
3440                                            MOVE    3,(2)                   ;*Fwt from indexed location
3441                                            CAME    3,[ZZ,,-1]              ;Test indexing
3442                                            STOP
3443
3444                                    ;**********
3445                                    >
3446
3447                                    ;This  test  verifies that index register addressing functions correctly.
3448                                    ;In this test,  a move  instruction using indexing is executed.  the move
3449                                    ;Instruction should place the contents of the location specified by index
3450                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3451                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3452                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3453
3454                    001601          SN=SN+1
3455                    000001                  ZZ=<ZZ+ZZ+1>&777777
3456                    777776                  IFE     <ZZ-1>,<ZZ=777776>
3457  032171  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3458  032172  525 01 0 00 777776                HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3459  032173  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3460  032174  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3461  032175  312 03 0 00 034466                CAME    3,[ZZ,,-1]              ;Test indexing
3462                                            STOP^
3463  032176  254 04 0 00 032177                HALT    .+1                     ;Test failed if program halts here
3464  032177  324 00 0 00 032200                JUMPA   .+1                     ;To loop change this instruction^
3465
3466                                    ;**********
3467
3468
3469                                    ;This  test  verifies that index register addressing functions correctly.
3470                                    ;In this test,  a move  instruction using indexing is executed.  the move
3471                                    ;Instruction should place the contents of the location specified by index
3472                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3473                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3474                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3475
3476                    001602          SN=SN+1
3477                    777775                  ZZ=<ZZ+ZZ+1>&777777
```

```
3478                                    IFE     <ZZ-1>,<ZZ=777776>
3479  032200  402 00 0 00 000003          SETZM   3                       ;Preload ac3 with 0
3480  032201  525 01 0 00 777775          HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3481  032202  201 02 0 00 000001          MOVEI   2,1                     ;Setup index register
3482  032203  200 03 0 02 000000          MOVE    3,(2)                   ;*Fwt from indexed location
3483  032204  312 03 0 00 034467          CAME    3,[ZZ,,-1]              ;Test indexing
3484                                      STOP^
3485  032205  254 04 0 00 032206          HALT    .+1                     ;Test failed if program halts here
3486  032206  324 00 0 00 032207          JUMPA   .+1                     ;To loop change this instruction^
3487
3488                                  ;**********
3489
3490
3491                                  ;This  test  verifies that index register addressing functions correctly.
3492                                  ;In this test,  a move  instruction using indexing is executed.  the move
3493                                  ;Instruction should place the contents of the location specified by index
3494                                  ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3495                                  ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3496                                  ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3497
3498                     001603       SN=SN+1
3499                     777773               ZZ=<ZZ+ZZ+1>&777777
3500                                          IFE     <ZZ-1>,<ZZ=777776>
3501  032207  402 00 0 00 000003          SETZM   3                       ;Preload ac3 with 0
3502  032210  525 01 0 00 777773          HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3503  032211  201 02 0 00 000001          MOVEI   2,1                     ;Setup index register
3504  032212  200 03 0 02 000000          MOVE    3,(2)                   ;*Fwt from indexed location
3505  032213  312 03 0 00 034470          CAME    3,[ZZ,,-1]              ;Test indexing
3506                                      STOP^
3507  032214  254 04 0 00 032215          HALT    .+1                     ;Test failed if program halts here
3508  032215  324 00 0 00 032216          JUMPA   .+1                     ;To loop change this instruction^
3509
3510                                  ;**********
3511
3512
3513                                  ;This  test  verifies that index register addressing functions correctly.
3514                                  ;In this test,  a move  instruction using indexing is executed.  the move
3515                                  ;Instruction should place the contents of the location specified by index
3516                                  ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3517                                  ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3518                                  ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3519
3520                     001604       SN=SN+1
3521                     777767               ZZ=<ZZ+ZZ+1>&777777
3522                                          IFE     <ZZ-1>,<ZZ=777776>
3523  032216  402 00 0 00 000003          SETZM   3                       ;Preload ac3 with 0
3524  032217  525 01 0 00 777767          HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3525  032220  201 02 0 00 000001          MOVEI   2,1                     ;Setup index register
3526  032221  200 03 0 02 000000          MOVE    3,(2)                   ;*Fwt from indexed location
3527  032222  312 03 0 00 034471          CAME    3,[ZZ,,-1]              ;Test indexing
3528                                      STOP^
3529  032223  254 04 0 00 032224          HALT    .+1                     ;Test failed if program halts here
3530  032224  324 00 0 00 032225          JUMPA   .+1                     ;To loop change this instruction^
3531
3532                                  ;**********
```

```
3533
3534
3535                                    ;This  test  verifies that index register addressing functions correctly.
3536                                    ;In this test,  a move  instruction using indexing is executed.  the move
3537                                    ;Instruction should place the contents of the location specified by index
3538                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3539                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3540                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3541
3542                    001605          SN=SN+1
3543                    777757                  ZZ=<ZZ+ZZ+1>&777777
3544                                            IFE     <ZZ-1>,<ZZ=777776>
3545  032225  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3546  032226  525 01 0 00 777757                HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3547  032227  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3548  032230  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3549  032231  312 03 0 00 034472                CAME    3,[ZZ,,-1]              ;Test indexing
3550                                            STOP^
3551  032232  254 04 0 00 032233                HALT    .+1                     ;Test failed if program halts here
3552  032233  324 00 0 00 032234                JUMPA   .+1                     ;To loop change this instruction^
3553
3554                                    ;**********
3555
3556
3557                                    ;This  test  verifies that index register addressing functions correctly.
3558                                    ;In this test,  a move  instruction using indexing is executed.  the move
3559                                    ;Instruction should place the contents of the location specified by index
3560                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3561                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3562                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3563
3564                    001606          SN=SN+1
3565                    777737                  ZZ=<ZZ+ZZ+1>&777777
3566                                            IFE     <ZZ-1>,<ZZ=777776>
3567  032234  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3568  032235  525 01 0 00 777737                HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3569  032236  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3570  032237  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3571  032240  312 03 0 00 034473                CAME    3,[ZZ,,-1]              ;Test indexing
3572                                            STOP^
3573  032241  254 04 0 00 032242                HALT    .+1                     ;Test failed if program halts here
3574  032242  324 00 0 00 032243                JUMPA   .+1                     ;To loop change this instruction^
3575
3576                                    ;**********
3577
3578
3579                                    ;This  test  verifies that index register addressing functions correctly.
3580                                    ;In this test,  a move  instruction using indexing is executed.  the move
3581                                    ;Instruction should place the contents of the location specified by index
3582                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3583                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3584                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3585
3586                    001607          SN=SN+1
3587                    777677                  ZZ=<ZZ+ZZ+1>&777777
```

```
3588                                        IFE     <ZZ-1>,<ZZ=777776>
3589  032243  402 00 0 00 000003            SETZM   3                       ;Preload ac3 with 0
3590  032244  525 01 0 00 777677            HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3591  032245  201 02 0 00 000001            MOVEI   2,1                     ;Setup index register
3592  032246  200 03 0 02 000000            MOVE    3,(2)                   ;*Fwt from indexed location
3593  032247  312 03 0 00 034474            CAME    3,[ZZ,,-1]              ;Test indexing
3594                                        STOP^
3595  032250  254 04 0 00 032251            HALT    .+1                     ;Test failed if program halts here
3596  032251  324 00 0 00 032252            JUMPA   .+1                     ;To loop change this instruction^
3597
3598                                  ;**********
3599
3600
3601                                  ;This  test  verifies that index register addressing functions correctly.
3602                                  ;In this test,  a move  instruction using indexing is executed.  the move
3603                                  ;Instruction should place the contents of the location specified by index
3604                                  ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3605                                  ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3606                                  ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3607
3608                    001610        SN=SN+1
3609                    777577              ZZ=<ZZ+ZZ+1>&777777
3610                                        IFE     <ZZ-1>,<ZZ=777776>
3611  032252  402 00 0 00 000003            SETZM   3                       ;Preload ac3 with 0
3612  032253  525 01 0 00 777577            HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3613  032254  201 02 0 00 000001            MOVEI   2,1                     ;Setup index register
3614  032255  200 03 0 02 000000            MOVE    3,(2)                   ;*Fwt from indexed location
3615  032256  312 03 0 00 034475            CAME    3,[ZZ,,-1]              ;Test indexing
3616                                        STOP^
3617  032257  254 04 0 00 032260            HALT    .+1                     ;Test failed if program halts here
3618  032260  324 00 0 00 032261            JUMPA   .+1                     ;To loop change this instruction^
3619
3620                                  ;**********
3621
3622
3623                                  ;This  test  verifies that index register addressing functions correctly.
3624                                  ;In this test,  a move  instruction using indexing is executed.  the move
3625                                  ;Instruction should place the contents of the location specified by index
3626                                  ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3627                                  ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3628                                  ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3629
3630                    001611        SN=SN+1
3631                    777377              ZZ=<ZZ+ZZ+1>&777777
3632                                        IFE     <ZZ-1>,<ZZ=777776>
3633  032261  402 00 0 00 000003            SETZM   3                       ;Preload ac3 with 0
3634  032262  525 01 0 00 777377            HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3635  032263  201 02 0 00 000001            MOVEI   2,1                     ;Setup index register
3636  032264  200 03 0 02 000000            MOVE    3,(2)                   ;*Fwt from indexed location
3637  032265  312 03 0 00 034476            CAME    3,[ZZ,,-1]              ;Test indexing
3638                                        STOP^
3639  032266  254 04 0 00 032267            HALT    .+1                     ;Test failed if program halts here
3640  032267  324 00 0 00 032270            JUMPA   .+1                     ;To loop change this instruction^
3641
3642                                  ;**********
```

```
3643
3644
3645                                    ;This  test  verifies that index register addressing functions correctly.
3646                                    ;In this test,  a move  instruction using indexing is executed.  the move
3647                                    ;Instruction should place the contents of the location specified by index
3648                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3649                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3650                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3651
3652                    001612          SN=SN+1
3653                    776777                  ZZ=<ZZ+ZZ+1>&777777
3654                                            IFE     <ZZ-1>,<ZZ=777776>
3655  032270  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3656  032271  525 01 0 00 776777                HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3657  032272  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3658  032273  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3659  032274  312 03 0 00 034477                CAME    3,[ZZ,,-1]              ;Test indexing
3660                                            STOP^
3661  032275  254 04 0 00 032276                HALT    .+1                     ;Test failed if program halts here
3662  032276  324 00 0 00 032277                JUMPA   .+1                     ;To loop change this instruction^
3663
3664                                    ;**********
3665
3666
3667                                    ;This  test  verifies that index register addressing functions correctly.
3668                                    ;In this test,  a move  instruction using indexing is executed.  the move
3669                                    ;Instruction should place the contents of the location specified by index
3670                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3671                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3672                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3673
3674                    001613          SN=SN+1
3675                    775777                  ZZ=<ZZ+ZZ+1>&777777
3676                                            IFE     <ZZ-1>,<ZZ=777776>
3677  032277  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3678  032300  525 01 0 00 775777                HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3679  032301  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3680  032302  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3681  032303  312 03 0 00 034500                CAME    3,[ZZ,,-1]              ;Test indexing
3682                                            STOP^
3683  032304  254 04 0 00 032305                HALT    .+1                     ;Test failed if program halts here
3684  032305  324 00 0 00 032306                JUMPA   .+1                     ;To loop change this instruction^
3685
3686                                    ;**********
3687
3688
3689                                    ;This  test  verifies that index register addressing functions correctly.
3690                                    ;In this test,  a move  instruction using indexing is executed.  the move
3691                                    ;Instruction should place the contents of the location specified by index
3692                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3693                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3694                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3695
3696                    001614          SN=SN+1
3697                    773777                  ZZ=<ZZ+ZZ+1>&777777
```

```
3698                                    IFE     <ZZ-1>,<ZZ=777776>
3699  032306  402 00 0 00 000003         SETZM   3                       ;Preload ac3 with 0
3700  032307  525 01 0 00 773777         HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3701  032310  201 02 0 00 000001         MOVEI   2,1                     ;Setup index register
3702  032311  200 03 0 02 000000         MOVE    3,(2)                   ;*Fwt from indexed location
3703  032312  312 03 0 00 034501         CAME    3,[ZZ,,-1]              ;Test indexing
3704                                    STOP^
3705  032313  254 04 0 00 032314         HALT    .+1                     ;Test failed if program halts here
3706  032314  324 00 0 00 032315         JUMPA   .+1                     ;To loop change this instruction^
3707
3708                                ;**********
3709
3710
3711                                ;This  test  verifies that index register addressing functions correctly.
3712                                ;In this test,  a move  instruction using indexing is executed.  the move
3713                                ;Instruction should place the contents of the location specified by index
3714                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3715                                ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3716                                ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3717
3718                     001615     SN=SN+1
3719                     767777             ZZ=<ZZ+ZZ+1>&777777
3720                                        IFE     <ZZ-1>,<ZZ=777776>
3721  032315  402 00 0 00 000003           SETZM   3                       ;Preload ac3 with 0
3722  032316  525 01 0 00 767777           HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3723  032317  201 02 0 00 000001           MOVEI   2,1                     ;Setup index register
3724  032320  200 03 0 02 000000           MOVE    3,(2)                   ;*Fwt from indexed location
3725  032321  312 03 0 00 034502           CAME    3,[ZZ,,-1]              ;Test indexing
3726                                      STOP^
3727  032322  254 04 0 00 032323           HALT    .+1                     ;Test failed if program halts here
3728  032323  324 00 0 00 032324           JUMPA   .+1                     ;To loop change this instruction^
3729
3730                                ;**********
3731
3732
3733                                ;This  test  verifies that index register addressing functions correctly.
3734                                ;In this test,  a move  instruction using indexing is executed.  the move
3735                                ;Instruction should place the contents of the location specified by index
3736                                ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3737                                ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3738                                ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3739
3740                     001616     SN=SN+1
3741                     757777             ZZ=<ZZ+ZZ+1>&777777
3742                                        IFE     <ZZ-1>,<ZZ=777776>
3743  032324  402 00 0 00 000003           SETZM   3                       ;Preload ac3 with 0
3744  032325  525 01 0 00 757777           HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3745  032326  201 02 0 00 000001           MOVEI   2,1                     ;Setup index register
3746  032327  200 03 0 02 000000           MOVE    3,(2)                   ;*Fwt from indexed location
3747  032330  312 03 0 00 034503           CAME    3,[ZZ,,-1]              ;Test indexing
3748                                      STOP^
3749  032331  254 04 0 00 032332           HALT    .+1                     ;Test failed if program halts here
3750  032332  324 00 0 00 032333           JUMPA   .+1                     ;To loop change this instruction^
3751
3752                                ;**********
```

```
3753
3754
3755                                    ;This  test  verifies that index register addressing functions correctly.
3756                                    ;In this test,  a move  instruction using indexing is executed.  the move
3757                                    ;Instruction should place the contents of the location specified by index
3758                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3759                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3760                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3761
3762                     001617         SN=SN+1
3763                     737777                 ZZ=<ZZ+ZZ+1>&777777
3764                                            IFE     <ZZ-1>,<ZZ=777776>
3765  032333  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3766  032334  525 01 0 00 737777                HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3767  032335  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3768  032336  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3769  032337  312 03 0 00 034504                CAME    3,[ZZ,,-1]              ;Test indexing
3770                                            STOP^
3771  032340  254 04 0 00 032341                HALT    .+1                     ;Test failed if program halts here
3772  032341  324 00 0 00 032342                JUMPA   .+1                     ;To loop change this instruction^
3773
3774                                    ;**********
3775
3776
3777                                    ;This  test  verifies that index register addressing functions correctly.
3778                                    ;In this test,  a move  instruction using indexing is executed.  the move
3779                                    ;Instruction should place the contents of the location specified by index
3780                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3781                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3782                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3783
3784                     001620         SN=SN+1
3785                     677777                 ZZ=<ZZ+ZZ+1>&777777
3786                                            IFE     <ZZ-1>,<ZZ=777776>
3787  032342  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3788  032343  525 01 0 00 677777                HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3789  032344  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3790  032345  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3791  032346  312 03 0 00 034505                CAME    3,[ZZ,,-1]              ;Test indexing
3792                                            STOP^
3793  032347  254 04 0 00 032350                HALT    .+1                     ;Test failed if program halts here
3794  032350  324 00 0 00 032351                JUMPA   .+1                     ;To loop change this instruction^
3795
3796                                    ;**********
3797
3798
3799                                    ;This  test  verifies that index register addressing functions correctly.
3800                                    ;In this test,  a move  instruction using indexing is executed.  the move
3801                                    ;Instruction should place the contents of the location specified by index
3802                                    ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3803                                    ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3804                                    ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3805
3806                     001621         SN=SN+1
3807                     577777                 ZZ=<ZZ+ZZ+1>&777777
```

```
3808                                            IFE     <ZZ-1>,<ZZ=777776>
3809  032351  402 00 0 00 000003                SETZM   3                       ;Preload ac3 with 0
3810  032352  525 01 0 00 577777                HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3811  032353  201 02 0 00 000001                MOVEI   2,1                     ;Setup index register
3812  032354  200 03 0 02 000000                MOVE    3,(2)                   ;*Fwt from indexed location
3813  032355  312 03 0 00 034506                CAME    3,[ZZ,,-1]              ;Test indexing
3814                                            STOP^
3815  032356  254 04 0 00 032357                HALT    .+1                     ;Test failed if program halts here
3816  032357  324 00 0 00 032360                JUMPA   .+1                     ;To loop change this instruction^
3817
3818                                  ;**********
3819
3820
3821                                  ;This  test  verifies that index register addressing functions correctly.
3822                                  ;In this test,  a move  instruction using indexing is executed.  the move
3823                                  ;Instruction should place the contents of the location specified by index
3824                                  ;Register 2 into ac3.  since  index register 2 was preloaded with a 1 and
3825                                  ;Ac1 was preloaded with a floating 0, the final result in ac3 should be a
3826                                  ;Floating 0.  if c(ac3)=a floating 0, this test passes.
3827
3828                    001622        SN=SN+1
3829                    377777                ZZ=<ZZ+ZZ+1>&777777
3830                                          IFE     <ZZ-1>,<ZZ=777776>
3831  032360  402 00 0 00 000003               SETZM   3                       ;Preload ac3 with 0
3832  032361  525 01 0 00 377777               HRLOI   1,ZZ                    ;Preload ac1 with floating 0
3833  032362  201 02 0 00 000001               MOVEI   2,1                     ;Setup index register
3834  032363  200 03 0 02 000000               MOVE    3,(2)                   ;*Fwt from indexed location
3835  032364  312 03 0 00 034507               CAME    3,[ZZ,,-1]              ;Test indexing
3836                                          STOP^
3837  032365  254 04 0 00 032366               HALT    .+1                     ;Test failed if program halts here
3838  032366  324 00 0 00 032367               JUMPA   .+1                     ;To loop change this instruction^
3839
3840                                  ;**********
3841
```

```
                                        ;Verify indexing where 'e' is non-zero
                                        
                    001700              SN=1700
          777777    777777                    ZZ=-1
          777777    777770                    XX=-10
                                        
                                        ;This test verifies that index register addressing functions  correctly.  in
                                        ;This test, the index reg is set-up with movei zz+1,zz-xx, where zz+1 is the
                                        ;Index reg. the ac is preloaded with its own address, 0,,zz. caie is used to
                                        ;Test the indexing operation. if the result in c(ac)=xx+c(zz+1 - right), this
                                        ;Test passes.  xx+c(zz+1 - right) should = zz.
                                        
                                        C1700:  REPEAT  ^D15,
                                        <SN=SN+1
                                                ZZ=ZZ+i
                                                XX=XX+3
                                                MOVEI   ZZ+1,ZZ-XX          ;Setup index register
                                                MOVEI   ZZ,ZZ               ;Preload ac with 0,,zz
                                                CAIE    ZZ,XX(ZZ+1)         ;*Test indexing
                                                STOP
                                        
                                        ;**********
                                        >
                    001701              SN=SN+1
                    000000                    ZZ=ZZ+1
          777777    777773                    XX=XX+3
032367    201 01 0 00 000005                  MOVEI   ZZ+1,ZZ-XX          ;Setup index register
032370    201 00 0 00 000000                  MOVEI   ZZ,ZZ               ;Preload ac with 0,,zz
032371    302 00 0 01 777773                  CAIE    ZZ,XX(ZZ+1)         ;*Test indexing
                                                STOP^
032372    254 04 0 00 032373                  HALT    .+1                 ;Test failed if program halts here
032373    324 00 0 00 032374                  JUMPA   .+1                 ;To loop change this instruction^
                                        
                                        ;**********
                                        
                    001702              SN=SN+1
                    000001                    ZZ=ZZ+1
          777777    777776                    XX=XX+3
032374    201 02 0 00 000003                  MOVEI   ZZ+1,ZZ-XX          ;Setup index register
032375    201 01 0 00 000001                  MOVEI   ZZ,ZZ               ;Preload ac with 0,,zz
032376    302 01 0 02 777776                  CAIE    ZZ,XX(ZZ+1)         ;*Test indexing
                                                STOP^
032377    254 04 0 00 032400                  HALT    .+1                 ;Test failed if program halts here
032400    324 00 0 00 032401                  JUMPA   .+1                 ;To loop change this instruction^
                                        
                                        ;**********
                                        
                    001703              SN=SN+1
                    000002                    ZZ=ZZ+1
                    000001                    XX=XX+3
032401    201 03 0 00 000001                  MOVEI   ZZ+1,ZZ-XX          ;Setup index register
032402    201 02 0 00 000002                  MOVEI   ZZ,ZZ               ;Preload ac with 0,,zz
032403    302 02 0 03 000001                  CAIE    ZZ,XX(ZZ+1)         ;*Test indexing
                                                STOP^
032404    254 04 0 00 032405                  HALT    .+1                 ;Test failed if program halts here
```

```
3897  032405  324 00 0 00 032406          JUMPA   .+1                     ;To loop change this instruction^
3898
3899                                      ;**********
3900
3901                  001704              SN=SN+1
3902                  000003                      ZZ=ZZ+1
3903                  000004                      XX=XX+3
3904  032406  201 04 0 00 777777          MOVEI   ZZ+1,ZZ-XX              ;Setup index register
3905  032407  201 03 0 00 000003          MOVEI   ZZ,ZZ                   ;Preload ac with 0,,zz
3906  032410  302 03 0 04 000004          CAIE    ZZ,XX(ZZ+1)             ;*Test indexing
3907                                      STOP^
3908  032411  254 04 0 00 032412          HALT    .+1                     ;Test failed if program halts here
3909  032412  324 00 0 00 032413          JUMPA   .+1                     ;To loop change this instruction^
3910
3911                                      ;**********
3912
3913                  001705              SN=SN+1
3914                  000004                      ZZ=ZZ+1
3915                  000007                      XX=XX+3
3916  032413  201 05 0 00 777775          MOVEI   ZZ+1,ZZ-XX              ;Setup index register
3917  032414  201 04 0 00 000004          MOVEI   ZZ,ZZ                   ;Preload ac with 0,,zz
3918  032415  302 04 0 05 000007          CAIE    ZZ,XX(ZZ+1)             ;*Test indexing
3919                                      STOP^
3920  032416  254 04 0 00 032417          HALT    .+1                     ;Test failed if program halts here
3921  032417  324 00 0 00 032420          JUMPA   .+1                     ;To loop change this instruction^
3922
3923                                      ;**********
3924
3925                  001706              SN=SN+1
3926                  000005                      ZZ=ZZ+1
3927                  000012                      XX=XX+3
3928  032420  201 06 0 00 777773          MOVEI   ZZ+1,ZZ-XX              ;Setup index register
3929  032421  201 05 0 00 000005          MOVEI   ZZ,ZZ                   ;Preload ac with 0,,zz
3930  032422  302 05 0 06 000012          CAIE    ZZ,XX(ZZ+1)             ;*Test indexing
3931                                      STOP^
3932  032423  254 04 0 00 032424          HALT    .+1                     ;Test failed if program halts here
3933  032424  324 00 0 00 032425          JUMPA   .+1                     ;To loop change this instruction^
3934
3935                                      ;**********
3936
3937                  001707              SN=SN+1
3938                  000006                      ZZ=ZZ+1
3939                  000015                      XX=XX+3
3940  032425  201 07 0 00 777771          MOVEI   ZZ+1,ZZ-XX              ;Setup index register
3941  032426  201 06 0 00 000006          MOVEI   ZZ,ZZ                   ;Preload ac with 0,,zz
3942  032427  302 06 0 07 000015          CAIE    ZZ,XX(ZZ+1)             ;*Test indexing
3943                                      STOP^
3944  032430  254 04 0 00 032431          HALT    .+1                     ;Test failed if program halts here
3945  032431  324 00 0 00 032432          JUMPA   .+1                     ;To loop change this instruction^
3946
3947                                      ;**********
3948
3949                  001710              SN=SN+1
3950                  000007                      ZZ=ZZ+1
3951                  000020                      XX=XX+3
```

```
3952  032432  201 10 0 00 777767          MOVEI   ZZ+1,ZZ-XX          ;Setup index register
3953  032433  201 07 0 00 000007          MOVEI   ZZ,ZZ               ;Preload ac with 0,,zz
3954  032434  302 07 0 10 000020          CAIE    ZZ,XX(ZZ+1)         ;*Test indexing
3955                                       STOP^
3956  032435  254 04 0 00 032436          HALT    .+1                 ;Test failed if program halts here
3957  032436  324 00 0 00 032437          JUMPA   .+1                 ;To loop change this instruction^
3958
3959                                 ;**********
3960
3961                 001711          SN=SN+1
3962                 000010                  ZZ=ZZ+1
3963                 000023                  XX=XX+3
3964  032437  201 11 0 00 777765          MOVEI   ZZ+1,ZZ-XX          ;Setup index register
3965  032440  201 10 0 00 000010          MOVEI   ZZ,ZZ               ;Preload ac with 0,,zz
3966  032441  302 10 0 11 000023          CAIE    ZZ,XX(ZZ+1)         ;*Test indexing
3967                                       STOP^
3968  032442  254 04 0 00 032443          HALT    .+1                 ;Test failed if program halts here
3969  032443  324 00 0 00 032444          JUMPA   .+1                 ;To loop change this instruction^
3970
3971                                 ;**********
3972
3973                 001712          SN=SN+1
3974                 000011                  ZZ=ZZ+1
3975                 000026                  XX=XX+3
3976  032444  201 12 0 00 777763          MOVEI   ZZ+1,ZZ-XX          ;Setup index register
3977  032445  201 11 0 00 000011          MOVEI   ZZ,ZZ               ;Preload ac with 0,,zz
3978  032446  302 11 0 12 000026          CAIE    ZZ,XX(ZZ+1)         ;*Test indexing
3979                                       STOP^
3980  032447  254 04 0 00 032450          HALT    .+1                 ;Test failed if program halts here
3981  032450  324 00 0 00 032451          JUMPA   .+1                 ;To loop change this instruction^
3982
3983                                 ;**********
3984
3985                 001713          SN=SN+1
3986                 000012                  ZZ=ZZ+1
3987                 000031                  XX=XX+3
3988  032451  201 13 0 00 777761          MOVEI   ZZ+1,ZZ-XX          ;Setup index register
3989  032452  201 12 0 00 000012          MOVEI   ZZ,ZZ               ;Preload ac with 0,,zz
3990  032453  302 12 0 13 000031          CAIE    ZZ,XX(ZZ+1)         ;*Test indexing
3991                                       STOP^
3992  032454  254 04 0 00 032455          HALT    .+1                 ;Test failed if program halts here
3993  032455  324 00 0 00 032456          JUMPA   .+1                 ;To loop change this instruction^
3994
3995                                 ;**********
3996
3997                 001714          SN=SN+1
3998                 000013                  ZZ=ZZ+1
3999                 000034                  XX=XX+3
4000  032456  201 14 0 00 777757          MOVEI   ZZ+1,ZZ-XX          ;Setup index register
4001  032457  201 13 0 00 000013          MOVEI   ZZ,ZZ               ;Preload ac with 0,,zz
4002  032460  302 13 0 14 000034          CAIE    ZZ,XX(ZZ+1)         ;*Test indexing
4003                                       STOP^
4004  032461  254 04 0 00 032462          HALT    .+1                 ;Test failed if program halts here
4005  032462  324 00 0 00 032463          JUMPA   .+1                 ;To loop change this instruction^
4006
```

```
4007                                    ;**********
4008
4009                    001715          SN=SN+1
4010                    000014                  ZZ=ZZ+1
4011                    000037                  XX=XX+3
4012  032463  201 15 0 00 777755                MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4013  032464  201 14 0 00 000014                MOVEI   ZZ,ZZ                   ;Preload ac with 0,,zz
4014  032465  302 14 0 15 000037                CAIE    ZZ,XX(ZZ+1)             ;*Test indexing
4015                                            STOP^
4016  032466  254 04 0 00 032467                HALT    .+1                     ;Test failed if program halts here
4017  032467  324 00 0 00 032470                JUMPA   .+1                     ;To loop change this instruction^
4018
4019                                    ;**********
4020
4021                    001716          SN=SN+1
4022                    000015                  ZZ=ZZ+1
4023                    000042                  XX=XX+3
4024  032470  201 16 0 00 777753                MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4025  032471  201 15 0 00 000015                MOVEI   ZZ,ZZ                   ;Preload ac with 0,,zz
4026  032472  302 15 0 16 000042                CAIE    ZZ,XX(ZZ+1)             ;*Test indexing
4027                                            STOP^
4028  032473  254 04 0 00 032474                HALT    .+1                     ;Test failed if program halts here
4029  032474  324 00 0 00 032475                JUMPA   .+1                     ;To loop change this instruction^
4030
4031                                    ;**********
4032
4033                    001717          SN=SN+1
4034                    000016                  ZZ=ZZ+1
4035                    000045                  XX=XX+3
4036  032475  201 17 0 00 777751                MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4037  032476  201 16 0 00 000016                MOVEI   ZZ,ZZ                   ;Preload ac with 0,,zz
4038  032477  302 16 0 17 000045                CAIE    ZZ,XX(ZZ+1)             ;*Test indexing
4039                                            STOP^
4040  032500  254 04 0 00 032501                HALT    .+1                     ;Test failed if program halts here
4041  032501  324 00 0 00 032502                JUMPA   .+1                     ;To loop change this instruction^
4042
4043                                    ;**********
4044
4045                                    PAGE
```

```
4046                    002000          SN=2000
4047            777777  777777                  ZZ=-1
4048            777777  777760                  XX=-20
4049
4050                                    ;This test verifies that index register  addressing functions correctly.  in
4051                                    ;This test, the index reg is set-up with movei zz+1,zz-xx, where zz+1 is the
4052                                    ;Index reg.  indexing  is  tested  by  loading the ac via movei zz,xx(zz+1),
4053                                    ;Where xx+c(zz+1)=zz. if the result in the ac equals 0,,zz, this test passes.
4054
4055                                    C2000:  REPEAT  ^D15,
4056                                    <SN=SN+1
4057                                            ZZ=ZZ+1
4058                                            XX=XX+5
4059                                            MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4060                                            MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4061                                            CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4062                                            STOP
4063
4064                                    ;**********
4065                                    >
4066                    002001          SN=SN+1
4067                    000000                  ZZ=ZZ+1
4068            777777  777765                  XX=XX+5
4069  032502    201 01 0 00 000013              MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4070  032503    201 00 0 01 777765              MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4071  032504    302 00 0 00 000000              CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4072                                            STOP^
4073  032505    254 04 0 00 032506              HALT    .+1                     ;Test failed if program halts here
4074  032506    324 00 0 00 032507              JUMPA   .+1                     ;To loop change this instruction^
4075
4076                                    ;**********
4077
4078                    002002          SN=SN+1
4079                    000001                  ZZ=ZZ+1
4080            777777  777772                  XX=XX+5
4081  032507    201 02 0 00 000007              MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4082  032510    201 01 0 02 777772              MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4083  032511    302 01 0 00 000001              CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4084                                            STOP^
4085  032512    254 04 0 00 032513              HALT    .+1                     ;Test failed if program halts here
4086  032513    324 00 0 00 032514              JUMPA   .+1                     ;To loop change this instruction^
4087
4088                                    ;**********
4089
4090                    002003          SN=SN+1
4091                    000002                  ZZ=ZZ+1
4092            777777  777777                  XX=XX+5
4093  032514    201 03 0 00 000003              MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4094  032515    201 02 0 03 777777              MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4095  032516    302 02 0 00 000002              CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4096                                            STOP^
4097  032517    254 04 0 00 032520              HALT    .+1                     ;Test failed if program halts here
4098  032520    324 00 0 00 032521              JUMPA   .+1                     ;To loop change this instruction^
4099
4100                                    ;**********
```

```
4101
4102                    002004          SN=SN+1
4103                    000003                  ZZ=ZZ+1
4104                    000004                  XX=XX+5
4105  032521  201 04 0 00 777777               MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4106  032522  201 03 0 04 000004               MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4107  032523  302 03 0 00 000003               CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4108                                           STOP^
4109  032524  254 04 0 00 032525               HALT    .+1                     ;Test failed if program halts here
4110  032525  324 00 0 00 032526               JUMPA   .+1                     ;To loop change this instruction^
4111
4112                                  ;**********
4113
4114                    002005          SN=SN+1
4115                    000004                  ZZ=ZZ+1
4116                    000011                  XX=XX+5
4117  032526  201 05 0 00 777773               MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4118  032527  201 04 0 05 000011               MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4119  032530  302 04 0 00 000004               CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4120                                           STOP^
4121  032531  254 04 0 00 032532               HALT    .+1                     ;Test failed if program halts here
4122  032532  324 00 0 00 032533               JUMPA   .+1                     ;To loop change this instruction^
4123
4124                                  ;**********
4125
4126                    002006          SN=SN+1
4127                    000005                  ZZ=ZZ+1
4128                    000016                  XX=XX+5
4129  032533  201 06 0 00 777767               MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4130  032534  201 05 0 06 000016               MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4131  032535  302 05 0 00 000005               CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4132                                           STOP^
4133  032536  254 04 0 00 032537               HALT    .+1                     ;Test failed if program halts here
4134  032537  324 00 0 00 032540               JUMPA   .+1                     ;To loop change this instruction^
4135
4136                                  ;**********
4137
4138                    002007          SN=SN+1
4139                    000006                  ZZ=ZZ+1
4140                    000023                  XX=XX+5
4141  032540  201 07 0 00 777763               MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4142  032541  201 06 0 07 000023               MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4143  032542  302 06 0 00 000006               CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4144                                           STOP^
4145  032543  254 04 0 00 032544               HALT    .+1                     ;Test failed if program halts here
4146  032544  324 00 0 00 032545               JUMPA   .+1                     ;To loop change this instruction^
4147
4148                                  ;**********
4149
4150                    002010          SN=SN+1
4151                    000007                  ZZ=ZZ+1
4152                    000030                  XX=XX+5
4153  032545  201 10 0 00 777757               MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4154  032546  201 07 0 10 000030               MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4155  032547  302 07 0 00 000007               CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
```

```
4156                                            STOP^
4157  032550  254 04 0 00 032551                HALT    .+1                     ;Test failed if program halts here
4158  032551  324 00 0 00 032552                JUMPA   .+1                     ;To loop change this instruction^
4159
4160                                    ;**********
4161
4162                    002011          SN=SN+1
4163                    000010                  ZZ=ZZ+1
4164                    000035                  XX=XX+5
4165  032552  201 11 0 00 777753                MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4166  032553  201 10 0 11 000035                MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4167  032554  302 10 0 00 000010                CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4168                                            STOP^
4169  032555  254 04 0 00 032556                HALT    .+1                     ;Test failed if program halts here
4170  032556  324 00 0 00 032557                JUMPA   .+1                     ;To loop change this instruction^
4171
4172                                    ;**********
4173
4174                    002012          SN=SN+1
4175                    000011                  ZZ=ZZ+1
4176                    000042                  XX=XX+5
4177  032557  201 12 0 00 777747                MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4178  032560  201 11 0 12 000042                MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4179  032561  302 11 0 00 000011                CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4180                                            STOP^
4181  032562  254 04 0 00 032563                HALT    .+1                     ;Test failed if program halts here
4182  032563  324 00 0 00 032564                JUMPA   .+1                     ;To loop change this instruction^
4183
4184                                    ;**********
4185
4186                    002013          SN=SN+1
4187                    000012                  ZZ=ZZ+1
4188                    000047                  XX=XX+5
4189  032564  201 13 0 00 777743                MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4190  032565  201 12 0 13 000047                MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4191  032566  302 12 0 00 000012                CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4192                                            STOP^
4193  032567  254 04 0 00 032570                HALT    .+1                     ;Test failed if program halts here
4194  032570  324 00 0 00 032571                JUMPA   .+1                     ;To loop change this instruction^
4195
4196                                    ;**********
4197
4198                    002014          SN=SN+1
4199                    000013                  ZZ=ZZ+1
4200                    000054                  XX=XX+5
4201  032571  201 14 0 00 777737                MOVEI   ZZ+1,ZZ-XX              ;Setup index register
4202  032572  201 13 0 14 000054                MOVEI   ZZ,XX(ZZ+1)             ;*Test indexing
4203  032573  302 13 0 00 000013                CAIE    ZZ,ZZ                   ;Pass if c(ac)=0,,address of ac
4204                                            STOP^
4205  032574  254 04 0 00 032575                HALT    .+1                     ;Test failed if program halts here
4206  032575  324 00 0 00 032576                JUMPA   .+1                     ;To loop change this instruction^
4207
4208                                    ;**********
4209
4210                    002015          SN=SN+1
```

```
4211                    000014                  ZZ=ZZ+1
4212                    000061                  XX=XX+5
4213  032576  201 15 0 00 777733                MOVEI   ZZ+1,ZZ-XX          ;Setup index register
4214  032577  201 14 0 15 000061                MOVEI   ZZ,XX(ZZ+1)         ;*Test indexing
4215  032600  302 14 0 00 000014                CAIE    ZZ,ZZ               ;Pass if c(ac)=0,,address of ac
4216                                            STOP^
4217  032601  254 04 0 00 032602                HALT    .+1                 ;Test failed if program halts here
4218  032602  324 00 0 00 032603                JUMPA   .+1                 ;To loop change this instruction^
4219
4220                                    ;**********
4221
4222                    002016          SN=SN+1
4223                    000015                  ZZ=ZZ+1
4224                    000066                  XX=XX+5
4225  032603  201 16 0 00 777727                MOVEI   ZZ+1,ZZ-XX          ;Setup index register
4226  032604  201 15 0 16 000066                MOVEI   ZZ,XX(ZZ+1)         ;*Test indexing
4227  032605  302 15 0 00 000015                CAIE    ZZ,ZZ               ;Pass if c(ac)=0,,address of ac
4228                                            STOP^
4229  032606  254 04 0 00 032607                HALT    .+1                 ;Test failed if program halts here
4230  032607  324 00 0 00 032610                JUMPA   .+1                 ;To loop change this instruction^
4231
4232                                    ;**********
4233
4234                    002017          SN=SN+1
4235                    000016                  ZZ=ZZ+1
4236                    000073                  XX=XX+5
4237  032610  201 17 0 00 777723                MOVEI   ZZ+1,ZZ-XX          ;Setup index register
4238  032611  201 16 0 17 000073                MOVEI   ZZ,XX(ZZ+1)         ;*Test indexing
4239  032612  302 16 0 00 000016                CAIE    ZZ,ZZ               ;Pass if c(ac)=0,,address of ac
4240                                            STOP^
4241  032613  254 04 0 00 032614                HALT    .+1                 ;Test failed if program halts here
4242  032614  324 00 0 00 032615                JUMPA   .+1                 ;To loop change this instruction^
4243
4244                                    ;**********
4245
```

```
                                        SUBTTL  TEST OF EXCH INSTRUCTION

                                        ;**********

                                        ;This test verifies that exch moves c(e) into ac and moves c(ac) into e.
                                        ;In this case, ac=e=0 and c(ac)=c(e).  hence, the final result in ac0
                                        ;Should be 0.  if c(ac)=0, the test passes.

032615  400 00 0 00 000000      C2100:  SETZ                            ;Preload ac,e with 0
032616  250 00 0 00 000000              EXCH                            ;*Exch should place 0 into ac0
032617  332 00 0 00 000000              SKIPE                           ;Pass if c(ac0)=0
                                        STOP^
032620  254 04 0 00 032621              HALT    .+1                     ;Test failed if program halts here
032621  324 00 0 00 032622              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that exch moves c(e) into ac and moves c(ac) into e.
                                        ;In this case, ac=e=-1,,-1 and c(ac)=c(e).  hence, the final result in
                                        ;Ac0 should be -1,,-1.  if c(ac)=-1,,-1, the test passes.

032622  474 00 0 00 000000      C2200:  SETO                            ;Preload ac,e with -1,,-1
032623  250 00 0 00 000000              EXCH                            ;*Exch should place -1,,-1 into ac0
032624  312 00 0 00 034443              CAME    [-1]                    ;Pass if c(ac0)=-1,,-1
                                        STOP^
032625  254 04 0 00 032626              HALT    .+1                     ;Test failed if program halts here
032626  324 00 0 00 032627              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
4275                                    ;This test verifies that exch moves c(e) into ac and moves c(ac) into e.
4276                                    ;In this case, c(ac)=-1,,0 and c(e)=0,,-1.  hence, the final result in
4277                                    ;The ac should be 0,,-1 and the result in e should be -1,,0.  if these
4278                                    ;Results occur, the test passes.
4279
4280  032627  205 00 0 00 777777      C2400:  MOVSI   -1                      ;Preload ac with -1,,0
4281  032630  201 01 0 00 777777              MOVEI   1,-1                    ;Preload e with 0,,-1
4282  032631  250 00 0 00 000001              EXCH    1                       ;*Exch should place 0,,-1 into ac and -1,,0 into e
4283  032632  312 01 0 00 034510              CAME    1,[-1,,0]               ;Pass if c(e)=-1,,0
4284                                          STOP^
4285  032633  254 04 0 00 032634              HALT    .+1                     ;Test failed if program halts here
4286  032634  324 00 0 00 032635              JUMPA   .+1                     ;To loop change this instruction^
4287  032635  312 00 0 00 034511      C2410:  CAME    0,[0,,-1]               ;Pass if c(ac)=0,,-1
4288                                          STOP^
4289  032636  254 04 0 00 032637              HALT    .+1                     ;Test failed if program halts here
4290  032637  324 00 0 00 032640              JUMPA   .+1                     ;To loop change this instruction^
4291
4292                                    ;**********
4293
4294                                    ;This test verifies that exch moves c(e) into ac and moves c(ac) into e.
4295                                    ;In this case, c(ac)=0,,-1 and c(e)=-1,,0.  hence, the final result in
4296                                    ;The ac should be -1,,0 and the result in e should be 0,,-1.  if these
4297                                    ;Results occur, the test passes.
4298
4299  032640  201 00 0 00 777777      C2700:  MOVEI   -1                      ;Preload ac with 0,,-1
4300  032641  205 01 0 00 777777              MOVSI   1,-1                    ;Preload e with -1,,0
4301  032642  250 00 0 00 000001              EXCH    1                       ;*Exch should place -1,,0 into ac and 0,,-1 into e
4302  032643  302 01 0 00 777777              CAIE    1,-1                    ;Pass if c(e)=0,,-1
4303                                          STOP^
4304  032644  254 04 0 00 032645              HALT    .+1                     ;Test failed if program halts here
4305  032645  324 00 0 00 032646              JUMPA   .+1                     ;To loop change this instruction^
4306  032646  312 00 0 00 034510      C2710:  CAME    ,[XWD -1,0]             ;Pass if c(ac)=-1,,0
4307                                          STOP^
4308  032647  254 04 0 00 032650              HALT    .+1                     ;Test failed if program halts here
4309  032650  324 00 0 00 032651              JUMPA   .+1                     ;To loop change this instruction^
4310
4311                                    ;**********
```

```
4312                                    ;This test is a reliability check of exch.  first, ac, e are preloaded
4313                                    ;With 252525,,252525. there, exch is executed 7 times.  the ac is then
4314                                    ;Checked for 252525,,252525.   if c(ac)=c(e)=252525,,252525, this test
4315                                    ;Passes.  in this test ac=e=ac0.
4316
4317  032651  200 00 0 00 034512       C3000:  MOVE    [252525252525]          ;Preload ac,e with 252525,,252525
4318                                            REPEAT  7,
4319                                    <       EXCH                            ;*Exch should place 252525,,252525 into ac0>
4320  032652  250 00 0 00 000000               EXCH                            ;*Exch should place 252525,,252525 into ac0
4321  032653  250 00 0 00 000000               EXCH                            ;*Exch should place 252525,,252525 into ac0
4322  032654  250 00 0 00 000000               EXCH                            ;*Exch should place 252525,,252525 into ac0
4323  032655  250 00 0 00 000000               EXCH                            ;*Exch should place 252525,,252525 into ac0
4324  032656  250 00 0 00 000000               EXCH                            ;*Exch should place 252525,,252525 into ac0
4325  032657  250 00 0 00 000000               EXCH                            ;*Exch should place 252525,,252525 into ac0
4326  032660  250 00 0 00 000000               EXCH                            ;*Exch should place 252525,,252525 into ac0
4327  032661  312 00 0 00 034512               CAME    [252525252525]          ;Pass if c(ac0)=252525,,252525
4328                                            STOP^
4329  032662  254 04 0 00 032663               HALT    .+1                     ;Test failed if program halts here
4330  032663  324 00 0 00 032664               JUMPA   .+1                     ;To loop change this instruction^
4331
4332                                    ;**********
4333
4334                                    ;This test verifies that exch moves c(ac) into e and c(e) into the ac.
4335                                    ;In this case, c(ac)=0 and c(e)=-1,,-1.  exch is executed 7 times; then,
4336                                    ;The ac is checked for -1,,-1 and e is checked for 0.  if either of these
4337                                    ;Results are not found, this test fails.
4338
4339  032664  400 00 0 00 000000       C3100:  SETZ                            ;Preload ac with 0
4340  032665  474 01 0 00 000000               SETO    1,0                     ;Preload e with -1,,-1
4341                                            REPEAT  7,
4342                                    <       EXCH    1                       ;*Exch should exchange c(ac) and c(e)>
4343  032666  250 00 0 00 000001               EXCH    1                       ;*Exch should exchange c(ac) and c(e)
4344  032667  250 00 0 00 000001               EXCH    1                       ;*Exch should exchange c(ac) and c(e)
4345  032670  250 00 0 00 000001               EXCH    1                       ;*Exch should exchange c(ac) and c(e)
4346  032671  250 00 0 00 000001               EXCH    1                       ;*Exch should exchange c(ac) and c(e)
4347  032672  250 00 0 00 000001               EXCH    1                       ;*Exch should exchange c(ac) and c(e)
4348  032673  250 00 0 00 000001               EXCH    1                       ;*Exch should exchange c(ac) and c(e)
4349  032674  250 00 0 00 000001               EXCH    1                       ;*Exch should exchange c(ac) and c(e)
4350  032675  312 00 0 00 034443               CAME    [-1]                    ;Pass if c(ac)=-1,,-1
4351                                            STOP^
4352  032676  254 04 0 00 032677               HALT    .+1                     ;Test failed if program halts here
4353  032677  324 00 0 00 032700               JUMPA   .+1                     ;To loop change this instruction^
4354  032700  312 01 0 00 034513       C3110:  CAME    1,[0]                   ;Pass if c(e)=0
4355                                            STOP^
4356  032701  254 04 0 00 032702               HALT    .+1                     ;Test failed if program halts here
4357  032702  324 00 0 00 032703               JUMPA   .+1                     ;To loop change this instruction^
```

```
4358                                    SUBTTL  TEST OF MOVEM INSTRUCTION
4359
4360                                    ;This test verifies that movem places c(ac) into e and does not modify c(ac)
4361                                    ;In this case, c(ac)=-1,,-1 and c(e)=0.  hence, the result in ac and e should
4362                                    ;Be -1,,-1.  if c(ac) and c(e)=-1,,-1, this test passes
4363
4364  032703  474 00 0 00 000000       C3200:  SETO                            ;Preload ac with -1,,-1
4365  032704  400 01 0 00 000000               SETZ    1,0                     ;Preload e with 0
4366  032705  202 00 0 00 000001               MOVEM   1                       ;*Movem should place -1,,-1 into e
4367  032706  312 01 0 00 034443               CAME    1,[-1]                  ;Pass if c(e)=-1,,-1
4368                                           STOP^
4369  032707  254 04 0 00 032710               HALT    .+1                     ;Test failed if program halts here
4370  032710  324 00 0 00 032711               JUMPA   .+1                     ;To loop change this instruction^
4371  032711  312 00 0 00 034443       C3210:  CAME    0,[-1]                  ;Pass if c(ac)=-1,,-1
4372                                           STOP^
4373  032712  254 04 0 00 032713               HALT    .+1                     ;Test failed if program halts here
4374  032713  324 00 0 00 032714               JUMPA   .+1                     ;To loop change this instruction^
4375
4376                                    ;**********
```

```
                                        SUBTTL  TEST OF JFCL INSTRUCTION AND ARITHMETIC FLAGS

                                        ;**********

                                        ;This test verifies that jfcl 17,.+1 always returns to the next sequential
                                        ;Instruction.  if jfcl skips the next instruction, this test fails.

032714  200 00 0 00 034514      C3300:  MOVE    [HALT .+3]              ;This instruction should not affect the test
032715  255 17 0 00 032716              JFCL    17,.+1                  ;*Jfcl should return to next sequential instruction
032716  304 00 0 00 000000              CAIA                            ;Skip halt instruction if jfcl passes
                                        STOP^
032717  254 04 0 00 032720              HALT    .+1                     ;Test failed if program halts here
032720  324 00 0 00 032721              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that  jfcl 17,.+1  always clears the cry0 flag.  addi is
                                        ;Used to set cry0.  then, jfcl 17,.+1 is executed to clear cry0.  jfcl 4,.+2
                                        ;Is executed to determine whether cry0 was reset by the previous jfcl.  this
                                        ;Test fails if jfcl 17,.+1 did not clear cry0.

032721  200 00 0 00 034443      C3400:  MOVE    [-1]                    ;Preload ac with -1,,-1
032722  271 00 0 00 000001              ADDI    1                       ;Set cry0 flag
032723  255 17 0 00 032724              JFCL    17,.+1                  ;*Clear arithmetic flags
032724  255 04 0 00 032726              JFCL    4,.+2                   ;Pass if cry0 was reset by previous instruction
032725  334 00 0 00 000000              SKIPA                           ;Skip halt if cry0 was cleared
                                        STOP^
032726  254 04 0 00 032727              HALT    .+1                     ;Test failed if program halts here
032727  324 00 0 00 032730              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                ;This test verifies that jfcl 17,.+1 always clears the cry1 flag.
                                ;Addi is used to set cry1.  then, jfcl 17,.+1 is executed to clear cry1.
                                ;Jfcl 4,.+2 is executed to determine whether cry1 was reset by the previous jfcl.
                                ;This test fails if jfcl 17,.+1 did not clear cry1.

032730  200 00 0 00 034443      C3500:  MOVE    [-1]                    ;Preload ac with -1,,-1
032731  271 00 0 00 000001              ADDI    1                       ;Set cry1 flag
032732  255 17 0 00 032733              JFCL    17,.+1                  ;*Clear arithmetic flags
032733  255 02 0 00 032735              JFCL    2,.+2                   ;Pass if cry1 was reset by previous instruction
032734  334 00 0 00 000000              SKIPA                           ;Skip halt if cry1 was cleared
                                        STOP^
032735  254 04 0 00 032736              HALT    .+1                     ;Test failed if program halts here
032736  324 00 0 00 032737              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that jfcl 17,.+1 always clears the arov flag.
                                ;Addi is used to set arov.  then, jfcl 17,.+1 is executed to clear arov.
                                ;Jfcl 4,.+2 is executed to determine whether arov was reset by the previous jfcl.
                                ;This test fails if jfcl 17,.+1 did not clear arov.

032737  205 00 0 00 400000      C3600:  MOVSI   400000                  ;Preload ac with -1,,-1
032740  270 00 0 00 034442              ADD     [XWD 400000,0]          ;Set arov flag
032741  255 17 0 00 032742              JFCL    17,.+1                  ;*Clear arithmetic flags
032742  255 10 0 00 032744              JFCL    10,.+2                  ;Pass if arov was reset by previous instruction
032743  334 00 0 00 000000              SKIPA                           ;Skip halt if arov was cleared
                                        STOP^
032744  254 04 0 00 032745              HALT    .+1                     ;Test failed if program halts here
032745  324 00 0 00 032746              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
                                        ;This test verifies that jfcl 0, is a no-op.
                                        ;In this test, add is used to set cry0.  then jfcl 0,.+2 is executed.
                                        ;If jfcl 0,.+2 does not skip the next instruction, this test passes.

032746  205 00 0 00 400000      C3700:  MOVSI   400000                  ;Preload ac with most negative number
032747  270 00 0 00 034443              ADD     [-1]                    ;Set cry0 flag
032750  255 00 0 00 032752              JFCL    .+2                     ;*Jfcl should return to next sequential instruction
032751  334 00 0 00 000000              SKIPA                           ;Pass if jfcl did not skip
                                        STOP^
032752  254 04 0 00 032753              HALT    .+1                     ;Test failed if program halts here
032753  324 00 0 00 032754              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that jfcl 0, is a no-op.
                                        ;In this test, add is used to set cry1.  then jfcl 0,.+2 is executed.
                                        ;If jfcl 0,.+2 does not skip the next instruction, this test passes.

032754  205 00 0 00 200000      C4000:  MOVSI   200000                  ;Preload ac with most negative number
032755  270 00 0 00 034441              ADD     [XWD 200000,0]          ;Set cry1 flag
032756  255 00 0 00 032760              JFCL    .+2                     ;*Jfcl should return to next sequential instruction
032757  334 00 0 00 000000              SKIPA                           ;Pass if jfcl did not skip
                                        STOP^
032760  254 04 0 00 032761              HALT    .+1                     ;Test failed if program halts here
032761  324 00 0 00 032762              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                        ;This test verifies that addi has the ability to set an arithmetic flag and
                                        ;That 'jfcl 17,,+2' jumps whenever any arithmetic flag is set.  in this test,
                                        ;Addi should set cry0 and cry1. then, jfcl should skip because a flag was set
                                        ;By addi.  if this test fails, addi could have  failed  to set a flag or jfcl
                                        ;Could have failed to jump when a flag was set.

032762  200 00 0 00 034443      C4100:  MOVE    [-1]                    ;Preload ac with all ones
032763  271 00 0 00 000001              ADDI    1                       ;*Addi should set cry0/1 flags
032764  255 17 0 00 032766              JFCL    17,,+2                  ;*Jfcl should jump because flags are set
                                        STOP^
032765  254 04 0 00 032766              HALT    .+1                     ;Test failed if program halts here
032766  324 00 0 00 032767              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that cai does not clear any arithmetic flags.
                                        ;First, cry0 and cry1 are set by addi; then cai is executed.
                                        ;Jfcl should jump because flags are set.  if jfcl does not jump,
                                        ;The flags were cleared by cai.  hence, cai failed.

032767  200 00 0 00 034443      C4200:  MOVE    [-1]                    ;Preload ac with -1,,-1
032770  271 00 0 00 000001              ADDI    1                       ;Set cyr0/1 flags
032771  300 17 0 00 000017              CAI     17,17                   ;*Cai should not clear flags
032772  255 17 0 00 032774              JFCL    17,,+2                  ;Pass if cai cleared flags
                                        STOP^
032773  254 04 0 00 032774              HALT    .+1                     ;Test failed if program halts here
032774  324 00 0 00 032775              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                        ;This test verifies that addi has the ability to set an arithmetic flag and
                                        ;That 'jfcl 17,,+2' jumps whenever any arithmetic flag is set. in this test,
                                        ;Addi should set cry0 and cry1. then jfcl should skip because a flag was set
                                        ;By addi.  if this test fails, addi could have failed to set a flag or  jfcl
                                        ;Could have failed to jump when a flag was set.

032775  200 00 0 00 034443      C4300:  MOVE    [-1]                    ;Preload ac with all ones
032776  271 00 0 00 000001              ADDI    1                       ;*Addi should set cry1 flags
032777  255 02 0 00 033001              JFCL    2,,+2                   ;*Jfcl should jump because cry1 flag is set
                                        STOP^
033000  254 04 0 00 033001              HALT    .+1                     ;Test failed if program halts here
033001  324 00 0 00 033002              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that addi has the ability to set an arithmetic flag and
                                        ;That 'jfcl 17,,+2' jumps whenever any arithmetic flag is set. in this test,
                                        ;Addi should set cry0 and cry1.  then, jfcl should skip because a flag was
                                        ;Set by addi.  if this test fails, addi could have failed to set a flag or
                                        ;Jfcl could have failed to jump when a flag was set.

033002  200 00 0 00 034443      C4400:  MOVE    [-1]                    ;Preload ac with all ones
033003  271 00 0 00 000001              ADDI    1                       ;*Addi should set cry0 flag
033004  255 04 0 00 033006              JFCL    4,,+2                   ;*Jfcl should jump because cry0 flag is set
                                        STOP^
033005  254 04 0 00 033006              HALT    .+1                     ;Test failed if program halts here
033006  324 00 0 00 033007              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                ;This test verifies that addi has the ability to set an arithmetic flag and
                                ;That 'jfcl 17,.+2' jumps whenever any arithmetic flag is set. because a flag
                                ;Was set by add.  if this test fails, addi could have failed to set a flag or
                                ;Jfcl could have failed to jump when a flag was set.

033007  205 00 0 00 400000      C4500:  MOVSI   400000                  ;Preload ac with all ones
033010  270 00 0 00 034442              ADD     [XWD 400000,0]          ;*Add should set arov flag
033011  255 10 0 00 033013              JFCL    10,.+2                  ;*Jfcl should jump because arov flag is set
                                        STOP^
033012  254 04 0 00 033013              HALT    .+1                     ;Test failed if program halts here
033013  324 00 0 00 033014              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that jfcl 17,.+1 always clears the floating overflow
                                ;Flag (fov).  first jfcl 17,.+1 is executed to clear fov.  then, jfcl 1,.+2
                                ;Is executed to determine whether fov was cleared.  if fov was clear, this
                                ;Test passes.

033014  255 17 0 00 033015      C4600:  JFCL    17,.+1                  ;*Clear arithmetic flags
033015  255 01 0 00 033017              JFCL    1,.+2                   ;Pass if fov was cleared
033016  334 00 0 00 000000              SKIPA                           ;Skip halt if test passed
                                        STOP^
033017  254 04 0 00 033020              HALT    .+1                     ;Test failed if program halts here
033020  324 00 0 00 033021              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
                                        ;This test verifies that jfcl 13, does not reset cry0.
                                        ;First cry0 and cry1 are set by addi; then, jfcl 13,.+2 is executed
                                        ;To clear all arithmetic flags except cry0.
                                        ;This test passes if jfcl 13,.+1 did not reset cry0.

033021  200 00 0 00 034443      C4700:  MOVE    [-1]                    ;Reload ac with -1,,-1
033022  271 00 0 00 000001              ADDI    1                       ;Set cry0/1
033023  255 13 0 00 033024              JFCL    13,.+1                  ;*Jfcl 13, should not reset cry0
033024  255 04 0 00 033026              JFCL    4,.+2                   ;Fail if cry 0 was reset
                                        STOP^
033025  254 04 0 00 033026              HALT    .+1                     ;Test failed if program halts here
033026  324 00 0 00 033027              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that jfcl 15, does not reset cry1.
                                        ;First cry0 and cry1 are set by addi; then, jfcl15,.+2 is executed
                                        ;To clear all arithmetic flags except cry1.
                                        ;This test passes if jfcl 15,.+1 did not reset cry1.

033027  200 00 0 00 034443      C5000:  MOVE    [-1]                    ;Preload ac with -1,,-1
033030  271 00 0 00 000001              ADDI    1                       ;Set cry0/1
033031  255 15 0 00 033032              JFCL    15,.+1                  ;*Jfcl15, should not reset cry0
033032  255 02 0 00 033034              JFCL    2,.+2                   ;Fail if cry1 was reset
                                        STOP^
033033  254 04 0 00 033034              HALT    .+1                     ;Test failed if program halts here
033034  324 00 0 00 033035              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
4580                                    ;This test verifies that jfcl 17, does not reset arov.
4581                                    ;First arov is set by add; then, jfcl 17,.+2 is executed
4582                                    ;To clear all arithmetic flags except arov.
4583                                    ;This test passes if jfcl 17,.+1 did not reset arov.
4584
4585  033035  205 00 0 00 400000  C5100:  MOVSI   400000                  ;Preload ac with -1,,-1
4586  033036  270 00 0 00 034442         ADD     [XWD 400000,0]          ;Set arov
4587  033037  255 07 0 00 033040         JFCL    7,.+1                   ;*Jfcl 17, should not reset arov
4588  033040  255 10 0 00 033042         JFCL    10,.+2                  ;Fail if arov was reset
4589                                     STOP^
4590  033041  254 04 0 00 033042         HALT    .+1                     ;Test failed if program halts here
4591  033042  324 00 0 00 033043         JUMPA   .+1                     ;To loop change this instruction^
4592
4593                                    ;**********
4594
4595                                    ;This test verifies that add of 0 to 0 will not set arov.
4596                                    ;First, all flags are reset, then 0 is added to 0 via add.
4597                                    ;Arov is then checked.  if arov is set, this test fails.
4598
4599  033043  255 17 0 00 033044  C5200:  JFCL    17,.+1                  ;Reset arithmetic flags
4600  033044  400 00 0 00 000000         SETZ                            ;Preload ac,e with 0
4601  033045  270 00 0 00 000000         ADD                             ;*Add should not set arov
4602  033046  255 10 0 00 033050         JFCL    10,.+2                  ;Pass if arov was reset
4603  033047  334 00 0 00 000000         SKIPA                           ;Skip halt if add passed
4604                                     STOP^
4605  033050  254 04 0 00 033051         HALT    .+1                     ;Test failed if program halts here
4606  033051  324 00 0 00 033052         JUMPA   .+1                     ;To loop change this instruction^
4607
4608                                    ;**********
```

```
                                ;This test verifies that add of 0 to 0 will not set cry0.
                                ;First, all flags are reset, then 0 is added to 0 via add.
                                ;Cry0 is then checked.  if cry0 is set, this test fails.

033052  400 00 0 00 000000      C5300:  SETZ                    ;Reset arithmetic flags
033053  255 17 0 00 033054              JFCL    17,,.+1         ;Preload ac,e with 0
033054  270 00 0 00 000000              ADD                     ;*Add should not set cry0
033055  255 04 0 00 033057              JFCL    4,,.+2          ;Pass if cry0 was reset
033056  334 00 0 00 000000              SKIPA                   ;Skip halt if add passed
                                        STOP^
033057  254 04 0 00 033060              HALT    .+1             ;Test failed if program halts here
033060  324 00 0 00 033061              JUMPA   .+1             ;To loop change this instruction^

                                ;**********

                                ;This test verifies that add of 0 to 0 will not set cry1.
                                ;First, all flags are reset, then 0 is added to 0 via add.
                                ;Cry1 is then checked.  if cry1 is set, this test fails.

033061  400 00 0 00 000000      C5400:  SETZ                    ;Reset arithmetic flags
033062  255 17 0 00 033063              JFCL    17,,.+1         ;Preload ac,e with 0
033063  270 00 0 00 000000              ADD                     ;*Add should not set cry1
033064  255 02 0 00 033066              JFCL    2,,.+2          ;Pass if cry1 was reset
033065  334 00 0 00 000000              SKIPA                   ;Skip halt if add passed
                                        STOP^
033066  254 04 0 00 033067              HALT    .+1             ;Test failed if program halts here
033067  324 00 0 00 033070              JUMPA   .+1             ;To loop change this instruction^

                                ;**********
```

```
                                        ;This test verifies that the 30x and the 31x instruction groups do not affect
                                        ;The arithmetic flags.  first, the flags are cleared and ac0 is cleared; then,
                                        ;Cai and  cam  are executed.  the flags are then checked.  if any of the flags
                                        ;Are set, this test fails and cai or cam is considered to have erroneously set
                                        ;The flags.

033070  255 17 0 00 033071      C5500:  JFCL    17,.+1                  ;Clear all flags
033071  400 00 0 00 000000              SETZ                            ;Clear ac,0
033072  300 00 0 00 000000              CAI                             ;*Cai should not set any arithmetic flag
033073  310 00 0 00 034443              CAM     [-1]                    ;*Cam should not set any arithmetic flag
033074  255 17 0 00 033076              JFCL    17,.+2                  ;Fail if any flag was set
033075  334 00 0 00 000000              SKIPA                           ;Skip halt if test passed
                                        STOP^
033076  254 04 0 00 033077              HALT    .+1                     ;Test failed if program halts here
033077  324 00 0 00 033100              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that the boolean instruction groups do not affect the
                                        ;Arithmetic flags.  first the flags are cleared and ac0 is cleared; then,
                                        ;Xor [0] and xor [-1] are executed.  the flags are then checked.  if any
                                        ;Of the flags are set, this test fails and xor is considered to have
                                        ;Erroneously set the flags.

033100  255 17 0 00 033101      C5600:  JFCL    17,.+1                  ;Clear all flags
033101  474 00 0 00 000000              SETO                            ;Clear ac,0
033102  430 00 0 00 034513              XOR     [0]                     ;*Xor should not set any arithmetic flag
033103  430 00 0 00 034443              XOR     [-1]                    ;*Xor should not set any arithmetic flag
033104  255 17 0 00 033106              JFCL    17,.+2                  ;Fail if any flag was set
033105  334 00 0 00 000000              SKIPA                           ;Skip halt if test passed
                                        STOP^
033106  254 04 0 00 033107              HALT    .+1                     ;Test failed if program halts here
033107  324 00 0 00 033110              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                ;This test verifies that the aobjx instruction group does do not affect the
                                ;Arithmetic flags.  first the flags are cleared and ac0 is cleared; then, aobjn
                                ;And aobjp are executed.  the flags are then checked.  if any of the flags are
                                ;Set, this test fails and aobjn or aobjp is considered to have erroneously set
                                ;The flags.

033110  474 00 0 00 000000      C5700:  SETO                    ;Clear all flags
033111  255 17 0 00 033112              JFCL    17,.+1          ;Clear ac,0
033112  253 00 0 00 033113              AOBJN   .+1             ;*Aobjn should not set any arithmetic arithmetic
033113  252 00 0 00 033114              AOBJP   .+1             ;*Aobjp should not set any arithmetic flag
033114  255 17 0 00 033116              JFCL    17,.+2          ;Fail if any flag was set
033115  334 00 0 00 000000              SKIPA                   ;Skip halt if tst passed
                                        STOP^
033116  254 04 0 00 033117              HALT    .+1             ;Test failed if program halts here
033117  324 00 0 00 033120              JUMPA   .+1             ;To loop change this instruction^

                                ;**********
```

```
                                        ;This test verifies that skip does not affect the flags.
                                        ;First, the arithmetic flags are cleared; then, skip is executed.
                                        ;If skip sets arov, cryo, cry1 or fov, this test fails.

033120  255 17 0 00 033121      C5701:  JFCL    17,.+1                  ;Clear all flags
033121  330 00 0 00 034443              SKIP    0,[-1]                  ;*Skip should not set any flags
033122  255 17 0 00 033124              JFCL    17,.+2                  ;Fail if any flag is set
033123  334 00 0 00 000000              SKIPA                           ;Pass if no flag is set
                                        STOP^
033124  254 04 0 00 033125              HALT    .+1                     ;Test failed if program halts here
033125  324 00 0 00 033126              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that jump does not affect the flags.
                                        ;First, the arithmetic flags are cleared; then, jump is executed.
                                        ;If jump sets arov, cryo, cry1 or fov, this test fails.

033126  255 17 0 00 033127      C5702:  JFCL    17,.+1                  ;Clear all flags
033127  320 00 0 00 034443              JUMP    0,[-1]                  ;*Jump should not set any flags
033130  255 17 0 00 033132              JFCL    17,.+2                  ;Fail if any flag is set
033131  334 00 0 00 000000              SKIPA                           ;Pass if no flag is set
                                        STOP^
033132  254 04 0 00 033133              HALT    .+1                     ;Test failed if program halts here
033133  324 00 0 00 033134              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                SUBTTL  TEST OF JRST INSTRUCTION AND ARITHMETIC FLAGS

                                ;**********

                                ;This test verifies that 'jrst, 0' does not set any flags.
                                ;First the arithmetic flags are reset; then, 'jrst 0,.+1' is executed
                                ;The arov,cry0 and cry1 flags are then checked. if any
                                ;Of these flags are set, this test fails

033134  255 17 0 00 033135      C6000:  JFCL    17,.+1          ;Reset all flags
033135  254 00 0 00 033136              JRST    .+1             ;*Jrst should not set any flags
033136  255 16 0 00 033140              JFCL    16,.+2          ;Pass if no flags are set
033137  334 00 0 00 000000              SKIPA                   ;Skip halt if jrst passes
                                        STOP^
033140  254 04 0 00 033141              HALT    .+1             ;Test failed if program halts here
033141  324 00 0 00 033142              JUMPA   .+1             ;To loop change this instruction^

                                ;**********

                                ;This test verifies that 'move 2,2' does not set any flags.
                                ;First, the arithmetic flags are reset; then, 'move 2,2' is executed.
                                ;The fov,arov,cry0 and cry1 flags are then checked.  if any
                                ;Of these flags are set, this test fails.

033142  255 17 0 00 033143      C6100:  JFCL    17,.+1          ;Reset all flags
033143  200 02 0 00 000002              MOVE    2,2             ;*Move should not set any flags
033144  255 17 0 00 033146              JFCL    17,.+2          ;Pass if no flags are set
033145  334 00 0 00 000000              SKIPA                   ;Skip halt if move passed
                                        STOP^
033146  254 04 0 00 033147              HALT    .+1             ;Test failed if program halts here
033147  324 00 0 00 033150              JUMPA   .+1             ;To loop change this instruction^

                                ;**********
```

```
                                        ;This test verifies that jrst 2,.+1(1) will set the arithmetic flag
                                        ;Specified in ac1.  in this case, ac1 specifies the cry0 flag.
                                        ;First, all flags are reset; then ac1 is set to specify cry0.
                                        ;Next, jrst 2,.+1(1) is executed to set cry0.  cry0 is then checked.  if
                                        ;Cry0 is set, this test passes.  otherwise, jrst 2,.+1 is faulty.

033150  255 17 0 00 033151      C6200:  JFCL    17,.+1                  ;Clear all flags
                                        SFLAG   CRY0                    ^;Set cry0 flag

033151  205 01 0 00 200000              MOVSI   1,CRY0
033152  255 17 0 00 033153              JFCL    17,.+1                  ;Reset all flags
033153  254 02 0 01 033154              JRST    2,.+1(1)                ;Set CRY0 flag
033154  255 04 0 00 033156              JFCL    4,.+2                   ;Pass if cry0 is set
                                        STOP^
033155  254 04 0 00 033156              HALT    .+1                     ;Test failed if program halts here
033156  324 00 0 00 033157              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that jrst 2,.+(1) will set the arithmetic flag
                                        ;Specified in ac1.  in this case, ac1 contain 0.  hence, jrst 2,.+1(1)
                                        ;Should reset the arithmetic flags.  first, the cry0/1, flags are set by addi;
                                        ;Then the flags should be cleared by jrst 2,.+1.  if cry1 was cleared, the
                                        ;Test passes.  otherwise, jrst 2,.+1 is faulty.

033157  200 00 0 00 034443      C6300:  MOVE    [-1]                    ;Preload ac0 with -1,,-1
033160  271 00 0 00 000001              ADDI    1                       ;Set cry0/1 flags
                                        SFLAG   0                       ^;Reset all arithmetic flags

033161  205 01 0 00 000000              MOVSI   1,0
033162  255 17 0 00 033163              JFCL    17,.+1                  ;Reset all flags
033163  254 02 0 01 033164              JRST    2,.+1(1)                ;Set 0 flag
033164  255 02 0 00 033166              JFCL    2,.+2                   ;Pass if cry1 is reset
033165  334 00 0 00 000000              SKIPA                           ;Skip halt instruction if test passed
                                        STOP^
033166  254 04 0 00 033167              HALT    .+1                     ;Test failed if program halts here
033167  324 00 0 00 033170              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
4790
4791                                    ;This test verifies that jrst 2,.+(1) will set the arithmetic flag
4792                                    ;Specified in ac1.  in this case, ac1 contain 0.  hence, jrst 2,.+1(1)
4793                                    ;Should reset the arithmetic flags.  first, the cry0/1, flags are set by add
4794                                    ;Then the flags should be cleared by jrst 2,.+1.  if arov was cleared, the
4795                                    ;Test passes.  otherwise, jrst 2,.+1 is faulty.
4796
4797  033170  205 00 0 00 400000        C6400:  MOVSI   400000                  ;Preload ac0 with -1,,-1
4798  033171  270 00 0 00 034443              ADD     [-1]                    ;Set cry0 and arov flags
4799                                          SFLAG   0                       ^;Reset all arithmetic flags
4800
4801  033172  205 01 0 00 000000              MOVSI   1,0
4802  033173  255 17 0 00 033174              JFCL    17,.+1                  ;Reset all flags
4803  033174  254 02 0 01 033175              JRST    2,.+1(1)                ;Set 0 flag
4804  033175  255 10 0 00 033177              JFCL    10,.+2                  ;Pass if arov is reset
4805  033176  334 00 0 00 000000              SKIPA                           ;Skip halt instruction if test passed
4806                                          STOP^
4807  033177  254 04 0 00 033200              HALT    .+1                     ;Test failed if program halts here
4808  033200  324 00 0 00 033201              JUMPA   .+1                     ;To loop change this instruction^
4809
4810                                    ;**********
4811
4812                                    ;This test verifies that jrst 2,.+1(1) will set the arithmetic flag
4813                                    ;Specified in ac1.  in this case, ac1 specifies the arov flag.
4814                                    ;First, all flags are reset; then ac1 set to specify arov.
4815                                    ;Next, jrst 2,.+1(1) is executed to set arov.  arov is then checked.  if
4816                                    ;Arov is set, this test passes.  otherwise, jrst 2,.+1 is faulty.
4817
4818  033201                            C6500:  SFLAG   AROV                    ^;Set arov flag
4819
4820  033201  205 01 0 00 400000              MOVSI   1,AROV
4821  033202  255 17 0 00 033203              JFCL    17,.+1                  ;Reset all flags
4822  033203  254 02 0 01 033204              JRST    2,.+1(1)                ;Set AROV flag
4823  033204  255 10 0 00 033206              JFCL    10,.+2                  ;Pass if arov was set
4824                                          STOP^
4825  033205  254 04 0 00 033206              HALT    .+1                     ;Test failed if program halts here
4826  033206  324 00 0 00 033207              JUMPA   .+1                     ;To loop change this instruction^
4827
4828                                    ;**********
```

```
                                        ;This test verifies that jrst 2,.+1(1) will set the arithmetic flag
                                        ;Specified in ac1.  in this case, ac1 specifies the cry1 flag.
                                        ;First, all flags are reset; then ac1 set to specify cry1.
                                        ;Next, jrst 2,.+1(1) is executed to set cry1.  cry1 is then checked.  if
                                        ;Cry1 is set, this test passes.  otherwise, jrst 2,.+1 is faulty.

033207                                  C6600:  SFLAG   CRY1                     ^;Set cry1 flag

033207  205 01 0 00 100000                      MOVSI   1,CRY1
033210  255 17 0 00 033211                      JFCL    17,.+1                  ;Reset all flags
033211  254 02 0 01 033212                      JRST    2,.+1(1)                ;Set CRY1 flag
033212  255 02 0 00 033214                      JFCL    2,.+2                   ;Pass if cry1 was set
                                                STOP^
033213  254 04 0 00 033214                      HALT    .+1                     ;Test failed if program halts here
033214  324 00 0 00 033215                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that jrst 2,.+1(1) will set the arithmetic flag
                                        ;Specified in ac1.  in this case, ac1 specifies the fov flag.
                                        ;First, all flags are reset; then ac1 set to specify fov.
                                        ;Next, jrst 2,.+1(1) is executed to set fov.  fov is then checked.  if
                                        ;Fov is set, this test passes.  otherwise, jrst 2,.+1 is faulty.

033215                                  C6700:  SFLAG   FOV                      ^;Set fov flag

033215  205 01 0 00 040000                      MOVSI   1,FOV
033216  255 17 0 00 033217                      JFCL    17,.+1                  ;Reset all flags
033217  254 02 0 01 033220                      JRST    2,.+1(1)                ;Set FOV flag
033220  255 01 0 00 033222                      JFCL    1,.+2                   ;Pass if fov was set
                                                STOP^
033221  254 04 0 00 033222                      HALT    .+1                     ;Test failed if program halts here
033222  324 00 0 00 033223                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                ;This test verifies that jfcl 0, should never jump.
                                ;First, fov is set via jrst2, ;then jfcl 0,
                                ;Is executed.  if jfcl 0, does not skip, this test passes.

033223                          C7000:  SFLAG   FOV                     ^;Set fov flag

033223  205 01 0 00 040000              MOVSI   1,FOV
033224  255 17 0 00 033225              JFCL    17,.+1                  ;Reset all flags
033225  254 02 0 01 033226              JRST    2,.+1(1)                ;Set FOV flag
033226  255 00 0 00 033230              JFCL    ,.+2                    ;*Jfcl should not jump
033227  334 00 0 00 000000              SKIPA                           ;Pass if jfcl did not jump
                                        STOP^
033230  254 04 0 00 033231              HALT    .+1                     ;Test failed if program halts here
033231  324 00 0 00 033232              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that  jrst 2,.+1(1)  will set the arithmetic  flag
                                ;Specified in ac1.  in this case, ac1 contains 0. hence, jrst 2,.+1(1)
                                ;Should  reset  the arithmetic flags.  first, the cry0, flag is set by
                                ;Jrst 2,.+1(1) with c(ac1)=cry0.  then, the flags should be cleared by
                                ;Jrst 2,.+1 with c(ac1)=0.  if cry0 was cleared, the test passes.
                                ;Otherwise, jrst 2,.+1 is faulty.

033232                          C7100:  SFLAG   CRY0                    ^;Set cry0 flags

033232  205 01 0 00 200000              MOVSI   1,CRY0
033233  255 17 0 00 033234              JFCL    17,.+1                  ;Reset all flags
033234  254 02 0 01 033235              JRST    2,.+1(1)                ;Set CRY0 flag
033235  400 01 0 00 000000              SETZ    1,                      ;Setup mask to clear arithmetic flags
033236  254 02 0 01 033237              JRST    2,.+1(1)                ;*Reset arithmetic flags
033237  255 04 0 00 033241              JFCL    4,.+2                   ;Pass if cry0 flag was reset
033240  334 00 0 00 000000              SKIPA                           ;Skip halt instruction if test pssed
                                        STOP^
033241  254 04 0 00 033242              HALT    .+1                     ;Test failed if program halts here
033242  324 00 0 00 033243              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
                                        ;This test verifies that jrst 2,.+1(1) will set the arithmetic flag
                                        ;Specified in ac1. in this case, ac1 contains 0. hence, jrst 2,.+1(1)
                                        ;Should reset the  arithmetic  flags.  first, the fov, flag is set by
                                        ;Jrst 2,.+1(1) with c(ac1)=fov.  then, the flags should be cleared by
                                        ;Jrst 2,.+1 with c(ac1)=0.  if fov was cleared, the test passes.
                                        ;Otherwise, jrst 2,.+1 is faulty.

033243                                  C7200:  SFLAG   FOV                     ^;Set fov flag

033243  205 01 0 00 040000                      MOVSI   1,FOV
033244  255 17 0 00 033245                      JFCL    17,.+1                  ;Reset all flags
033245  254 02 0 01 033246                      JRST    2,.+1(1)                ;Set FOV flag
033246  400 01 0 00 000000                      SETZ    1,                      ;Setup mask to clear arithmetic flags,
033247  254 02 0 01 033250                      JRST    2,.+1(1)                ;*Reset arithmetic flags
033250  255 01 0 00 033252                      JFCL    1,.+2                   ;Pass if fov flag was reset
033251  334 00 0 00 000000                      SKIPA                           ;Skip halt instruction if test pssed
                                                STOP^
033252  254 04 0 00 033253                      HALT    .+1                     ;Test failed if program halts here
033253  324 00 0 00 033254                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that  jrst 2,.+1(1)  will set the arithmetic flag
                                        ;Specified in ac1.  in this case, ac1 contains 0. hence, jrst 2,.+1(1)
                                        ;Should reset the arithmetic flags.  first, the arithmetic flags are
                                        ;Reset by jfcl 17,.+1.  then, the flags should be cleared by jrst 2,.+1.
                                        ;If all the arithmetic flags were cleared, the test passes.  otherwise,
                                        ;Jrst 2,.+1 is faulty.

033254  255 17 0 00 033255              C7300:  JFCL    17,.+1                  ;Clear flags
033255  400 01 0 00 000000                      SETZ    1,                      ;Setup mask to clear aritmetic flags
033256  254 02 0 01 033257                      JRST    2,.+1(1)                ;*Reset arithmetic flags
033257  255 17 0 00 033261                      JFCL    17,.+2                  ;Pass if all flags are reset
033260  334 00 0 00 000000                      SKIPA                           ;Skip halt instruction if test passed
                                                STOP^
033261  254 04 0 00 033262                      HALT    .+1                     ;Test failed if program halts here
033262  324 00 0 00 033263                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                SUBTTL  TEST OF JSP INSTRUCTION

                                ;**********

                                ;This test verifies that jsp always stores the flags and pc in the ac.
                                ;In this case, the flags are reset; then, jsp is executed.  the ac is then
                                ;Checked for its contents non-zero.  if c(ac)=0, it indicates
                                ;That neither the flags nor the pc was saved.  hence, this test fails.

033263  403 00 0 00 000001      C7400:  SETZB   1                       ;Clear ac and setup mask to reset flags
033264  254 02 0 01 033265              JRST    2,,+1(1)                ;Reset flags
033265  265 00 0 00 033266              JSP     .+1                     ;*Jsp should store flags and pc in the ac
033266  336 00 0 00 000000              SKIPN                           ;Pass if c(ac) is non-zero
                                        STOP                            ;It did not store any flags or pc

033267  254 04 0 00 033270              HALT    .+1                     ;Test failed if program halts here
033270  324 00 0 00 033271              JUMPA   .+1                     ;To loop change this instruction

                                ;**********

                                ;This test verifies that jsp always stores the pc in the right half of the ac.
                                ;In this case, the ac is cleared, then, jsp is executed.  the right half of
                                ;The ac is then checked for its contents non-zero.  if c(ac right half)
                                ;Is non-zero, it indicated that the pc was saved; and this test passes.

033271  400 00 0 00 000000      C7500:  SETZ                            ;Clean ac
033272  265 00 0 00 033273              JSP     .+1                     ;*Jsp should store the pc in the ac
033273  606 00 0 00 777777              TRNN    -1                      ;Pass if c(ac) in non-zero
                                        STOP^
033274  254 04 0 00 033275              HALT    .+1                     ;Test failed if program halts here
033275  324 00 0 00 033276              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
                                ;This test verifies that jsp always stores the flags in the left half of the ac.
                                ;Fist, the ac is cleared; then, some flags are set and jsp is executed.
                                ;The left half of the ac is checked for its contents non-zero to determine
                                ;Whether the flags were saved.  if c(ac-left) is non-zero, this test passes.

033276  402 00 0 00 000000      C7600:  SETZM   0                       ;Clear ac
033277  205 01 0 00 740000              MOVSI   1,740000                ;Set up mask to set flags
033300  254 02 0 01 033301              JRST    2,.+1(1)                ;Set some arithmetic flags
033301  265 00 0 00 033302              JSP     .+1                     ;*Jsp should store flags in the ac
033302  607 00 0 00 777777              TLNN    -1                      ;Pass if c(ac) is non-zero
                                        STOP^
033303  254 04 0 00 033304              HALT    .+1                     ;Test failed if program halts here
033304  324 00 0 00 033305              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that jsp always stores the flags in the left half of the
                                ;Ac. first, the ac is cleared; then, the arov flag is set and jsp is executed.
                                ;Then, the arov flag bit of the left half of the ac is checked for its contents
                                ;Non-zero to determine whether the arov flag was saved.  if the arov flag bit
                                ;Of the ac is set, this test passes.

033305  402 00 0 00 000000      C7700:  SETZM   0                       ;Clear the ac
                                        SFLAG   AROV                    ^;Set arov flag

033306  205 01 0 00 400000              MOVSI   1,AROV
033307  255 17 0 00 033310              JFCL    17,.+1                  ;Reset all flags
033310  254 02 0 01 033311              JRST    2,.+1(1)                ;Set AROV flag
033311  265 00 0 00 033312              JSP     .+1                     ;*Jsp should save the flags in the ac-left
033312  607 00 0 00 400000              TLNN    AROV                    ;Pass if arov was saved
                                        STOP^
033313  254 04 0 00 033314              HALT    .+1                     ;Test failed if program halts here
033314  324 00 0 00 033315              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
5009                                    ;This test verifies that jsp always stores the flags in the left half of the
5010                                    ;Ac. first, the ac is cleared; then, the cry0 flag is set and jsp is executed.
5011                                    ;Then, the cry0 flag bit of the left half of the ac is checked for its contents
5012                                    ;Non-zero to determine whether the cry0 flag was saved.  if the cry0 flag bit
5013                                    ;Of the ac is set, this test passes.
5014
5015  033315  402 00 0 00 000000        C10000: SETZM   0                       ;Clear the ac
5016                                            SFLAG   CRY0                    ^;Set cry0 flag
5017
5018  033316  205 01 0 00 200000                MOVSI   1,CRY0
5019  033317  255 17 0 00 033320                JFCL    17,.+1                  ;Reset all flags
5020  033320  254 02 0 01 033321                JRST    2,.+1(1)                ;Set CRY0 flag
5021  033321  265 00 0 00 033322                JSP     .+1                     ;*Jsp should save the flags in the ac-left
5022  033322  607 00 0 00 200000                TLNN    CRY0                    ;Pass if cry0 was saved
5023                                            STOP^
5024  033323  254 04 0 00 033324                HALT    .+1                     ;Test failed if program halts here
5025  033324  324 00 0 00 033325                JUMPA   .+1                     ;To loop change this instruction^
5026
5027                                    ;**********
5028
5029                                    ;This test verifies that jsp always stores the flags in the left half of the
5030                                    ;Ac. first, the ac is cleared; then, the cry1 flag is set and jsp is executed.
5031                                    ;Then, the cry1 flag bit of the left half of the ac is checked for its contents
5032                                    ;Non-zero to determine whether the flag was saved.  if the cry1 flag bit of the
5033                                    ;Ac is set, this test passes.
5034
5035  033325  400 00 0 00 000000        C10100: SETZ                            ;Clear ac
5036                                            SFLAG   CRY1                    ^;Set cry1 flag
5037
5038  033326  205 01 0 00 100000                MOVSI   1,CRY1
5039  033327  255 17 0 00 033330                JFCL    17,.+1                  ;Reset all flags
5040  033330  254 02 0 01 033331                JRST    2,.+1(1)                ;Set CRY1 flag
5041  033331  265 00 0 00 033332                JSP     .+1                     ;*Jsp should save the flags in the
5042  033332  607 00 0 00 100000                TLNN    CRY1                    ;Pass if arov was saved
5043                                            STOP^
5044  033333  254 04 0 00 033334                HALT    .+1                     ;Test failed if program halts here
5045  033334  324 00 0 00 033335                JUMPA   .+1                     ;To loop change this instruction^
5046
5047                                    ;**********
```

```
                                ;This test verifies that jsp always stores the flags in the left half of the
                                ;Ac.  first, the ac is cleared; then, the fov flag is set and jsp is executed.
                                ;Then, the fov flag b t of the left half of the ac is checked for its contents
                                ;Non-zero to determine whether the fov flag was saved.  if the fov flag bit of
                                ;The ac is set, this test passes.

033335  400 00 0 00 000000      C10200: SETZ                            ;Clear the ac
                                        SFLAG   FOV                     ^;Set fov flag

033336  205 01 0 00 040000              MOVSI   1,FOV
033337  255 17 0 00 033340              JFCL    17,.+1                  ;Reset all flags
033340  254 02 0 01 033341              JRST    2,.+1(1)                ;Set FOV flag
033341  265 00 0 00 033342              JSP     .+1                     ;*Jsp should save the flags in the ac-left
033342  607 00 0 00 040000              TLNN    FOV                     ;Pass if fov was saved
                                        STOP^
033343  254 04 0 00 033344              HALT    .+1                     ;Test failed if program halts here
033344  324 00 0 00 033345              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that jsp will store only flags that exist.  i.e. reset
                                ;Flags  will not be set in the ac.  only flags that are clearable by jrstf
                                ;Will be checked here.  first all clearable flags are reset by jrstf. then
                                ;Jsp is executed.  the ac is then checked.  if any clearable flags are set
                                ;In the ac, this test fails.

033345                          C10300: SFLAG   0                       ^;Clear all clearable flags

033345  205 01 0 00 000000              MOVSI   1,0
033346  255 17 0 00 033347              JFCL    17,.+1                  ;Reset all flags
033347  254 02 0 01 033350              JRST    2,.+1(1)                ;Set 0 flag
033350  265 00 0 00 033351              JSP     .+1                     ;*Jsp should not store clearable falgs
033351  603 00 0 00 761777              TLNE    761777                  ;Fail if any clearable flag is set
                                        STOP^
033352  254 04 0 00 033353              HALT    .+1                     ;Test failed if program halts here
033353  324 00 0 00 033354              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that jsp always jumps.
                                ;In this test, jsp .+2 is executed.  if jsp jumps, the
                                ;Test passes.  otherwise, this test halts.

033354  265 00 0 00 033356      C10400: JSP     .+2                     ;*Jsp should always jump
                                        STOP^
033355  254 04 0 00 033356              HALT    .+1                     ;Test failed if program halts here
033356  324 00 0 00 033357              JUMPA   .+1                     ;To loop change this instruction^
```

```
 5095                                   SUBTTL  TEST JRST INSTRUCTION
 5096
 5097                                   ;**********
 5098
 5099                                   ;This test verifies that jrst always jumps.
 5100                                   ;In this test, jrst .+2 is executed.  if jrst jumps, the test passes;
 5101                                   ;Otherwise, this test halts.
 5102
 5103  033357  254 00 0 00 033361       C10500: JRST    .+2                    ;*Jrst 0, should always jump
 5104                                           STOP^
 5105  033360  254 04 0 00 033361               HALT    .+1                    ;Test failed if program halts here
 5106  033361  324 00 0 00 033362               JUMPA   .+1                    ;To loop change this instruction^
 5107
 5108                                   ;**********
```

```
                                        SUBTTL  TEST OF AOBJX INSTRUCTIONS

                                        ;**********

                                        ;This test verifies that aobjn always adds 1 to both halves of the ac.
                                        ;First, the ac is cleared; then, aobjn is executed and the ac is checked
                                        ;For 1,,1.  if c(ac)=1,,1, this test passes.

033362  400 00 0 00 000000      C11200: SETZ                            ;Clear the ac
033363  253 00 0 00 033364              AOBJN   .+1                     ;*Aobjn should add 1 to both halves of the ac
033364  312 00 0 00 034417              CAME    [XWD 1,1]               ;Pass if c(ac)=1,,1
                                        STOP^
033365  254 04 0 00 033366              HALT    .+1                     ;Test failed if program halts here
033366  324 00 0 00 033367              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that aobjp always adds 1 to both halves of the ac.
                                        ;First, the ac is cleared; then aobjp is executed and the ac is checked
                                        ;For 1,,1.  if c(ac)=1,,1, this test passes.

033367  200 00 0 00 034515      C11300: MOVE    [XWD 377777,377777]     ;Preload ac with 377777,,377777
033370  252 00 0 00 033371              AOBJP   .+1                     ;*Aobjp should add, to both halves of the ac
033371  312 00 0 00 034516              CAME    [XWD 400000,400000]     ;Pass if c(ac)=400000,400000
                                        STOP^
033372  254 04 0 00 033373              HALT    .+1                     ;Test failed if program halts here
033373  324 00 0 00 033374              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                ;This test verifies that aobjn will not jump when c(ac) is positive
                                ;First, the ac is cleared; and aobjn is executed.  aobjn should not jump
                                ;Because c(ac) is positive.  if aobjn jumps, this test fails.

033374  400 00 0 00 000000      C11400: SETZ                    ;Clear the ac
033375  253 00 0 00 033377              AOBJN   .+2             ;*Aobjn should not jump when c(ac) is positive
033376  334 00 0 00 000000              SKIPA                   ;Pass if aobjn does not jump
                                        STOP^
033377  254 04 0 00 033400              HALT    .+1             ;Test failed if program halts here
033400  324 00 0 00 033401              JUMPA   .+1             ;To loop change this instruction^

                                ;**********

                                ;This test verifies that aobjp will jump when c(ac) is positive
                                ;First, the ac is cleared; and aobjp is executed.  aobjp should jump
                                ;Because c(ac) is positive.  if aobjp does not jump, this test fails.

033401  400 00 0 00 000000      C11500: SETZ                    ;Clear the ac
033402  252 00 0 00 033404              AOBJP   .+2             ;*Aobjp should jump because c(ac) is positive
                                        STOP^
033403  254 04 0 00 033404              HALT    .+1             ;Test failed if program halts here
033404  324 00 0 00 033405              JUMPA   .+1             ;To loop change this instruction^

                                ;**********
```

```
5162                                    ;This test verifies that aobjn will jump when c(ac) is negative.  first,
5163                                    ;The ac is preloaded with 400000,,0; and aobjn is executed.  aobjn should
5164                                    ;Jump because c(ac) is negative.  if aobjn does not jump, this test fails.
5165
5166  033405  200 00 0 00 034516        C11600: MOVE    [XWD 400000,400000]     ;Preload ac with 400000,,400000
5167  033406  253 00 0 00 033410                AOBJN   .+2                     ;*Aobjn should jump because c(ac) is negative
5168                                            STOP^
5169  033407  254 04 0 00 033410                HALT    .+1                     ;Test failed if program halts here
5170  033410  324 00 0 00 033411                JUMPA   .+1                     ;To loop change this instruction^
5171
5172                                    ;**********
5173
5174                                    ;This test verifies that aobjp will not jump when c(ac) is negative.  first,
5175                                    ;The ac is preloaded with 400000,,0; and aobjp is executed. aobjp should not
5176                                    ;Jump because c(ac) is negative.  if aobjp jumps, this test fails.
5177
5178  033411  200 00 0 00 034516        C11700: MOVE    [XWD 400000,400000]     ;Preload ac with 400000,,400000
5179  033412  252 00 0 00 033414                AOBJP   .+2                     ;*Aobjp should not jump because c(ac) is negative
5180  033413  334 00 0 00 000000                SKIPA                           ;Pass if aobjp does not jump
5181                                            STOP^
5182  033414  254 04 0 00 033415                HALT    .+1                     ;Test failed if program halts here
5183  033415  324 00 0 00 033416                JUMPA   .+1                     ;To loop change this instruction^
5184
5185                                    ;**********
5186
5187                                    ;This test verifies that there is no carry from bit 18 to bit 17 of the ac
5188                                    ;On aobjn. the ac is preloaded with -1,,-1; then aobjn is executed. aobjn
5189                                    ;Should add one to both halves of the ac without generating a carry from
5190                                    ;Bit 18 to bit 17. if no carry was generated, this test passes.
5191
5192  033416  4[illegible] 00 0 00 000000   C12000: SETO                            ;Preload ac with -1,, -1
5193  033417  253 00 0 00 033420                AOBJN   .+1                     ;*Aobjn should not cause a carry from bit 18 to 17
5194  033420  332 00 0 00 000000                SKIPE                           ;Pass if c(ac)=0 (no carry was generated)
5195                                            STOP^
5196  033421  254 04 0 00 033422                HALT    .+1                     ;Test failed if program halts here
5197  033422  324 00 0 00 033423                JUMPA   .+1                     ;To loop change this instruction^
5198
5199                                    ;**********
```

```
                                SUBTTL  TEST SETTING OF ARITHMETIC FLAGS VIA MOVNX AND MOVMX

                                ;This test verifies that movni sets cry0 and cry1 flags only when the data
                                ;Is 0. first, the arithmetic flags are reset; then, movni is executed with
                                ;Data of zeros.  the arithmetic flags are checked.  cry0/1 are set and arov
                                ;And fov reset, this test passes.

033423  255 17 0 00 033424      C12100: JFCL    17,,.+1                 ;Clear flags
033424  211 00 0 00 000000              MOVNI   0                       ;* Movni 0 should set cry0/1
033425  255 04 0 00 033427              JCRY0   .+2                     ;Pass if cry0 is set
                                        STOP^
033426  254 04 0 00 033427              HALT    .+1                     ;Test failed if program halts here
033427  324 00 0 00 033430              JUMPA   .+1                     ;To loop change this instruction^
033430  255 02 0 00 033432              JCRY1   .+2                     ;Pass if cry1 is set
                                        STOP^
033431  254 04 0 00 033432              HALT    .+1                     ;Test failed if program halts here
033432  324 00 0 00 033433              JUMPA   .+1                     ;To loop change this instruction^
033433  255 10 0 00 033435              JOV     .+2                     ;Pass if arov reset
033434  334 00 0 00 000000              SKIPA
                                        STOP^
033435  254 04 0 00 033436              HALT    .+1                     ;Test failed if program halts here
033436  324 00 0 00 033437              JUMPA   .+1                     ;To loop change this instruction^
033437  255 01 0 00 033441              JFOV    .+2                     ;Pass if fov reset
033440  334 00 0 00 000000              SKIPA
                                        STOP^
033441  254 04 0 00 033442              HALT    .+1                     ;Test failed if program halts here
033442  324 00 0 00 033443              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that movn will not set arithmetic flags when the data
                                ;Is -1,,-1.  first, the flags are reset; then, movn [-1,,-1] is executed.
                                ;The arithmetic flags are checked.  if  any arithmetic  flag is set, this
                                ;Test fails.

033443  255 17 0 00 033444      C12200: JFCL    17,,.+1                 ;Clear flags
033444  210 00 0 00 034443              MOVN    [-1]                    ;*Movn [-1,,-1] should set arithmetic flags
033445  255 17 0 00 033447              JFCL    17,,.+2                 ;Fail if an arithmetic flag is set
033446  334 00 0 00 000000              SKIPA                           ;Skip halt instruction if movn passed.
                                        STOP^
033447  254 04 0 00 033450              HALT    .+1                     ;Test failed if program halts here
033450  324 00 0 00 033451              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
                                ;This test verifies that movn will set the arov and
                                ;Cry1 flags only when the data is 400000,,0
                                ;First, the flags are reset; then, movn [400000,,0] is executed.
                                ;The arithmetic flags are checked.
                                ;If arov and cry1 are set and cry0 and fov are reset, this test passes.

033451  255 17 0 00 033452      C12300: JFCL    17,.+1                  ;Clear arithmetic flags
033452  210 00 0 00 034442              MOVN    [XWD 400000,0]          ;*Movn [400000,,0] should set arov and cry1 only
033453  255 10 0 00 033455              JOV     .+2                     ;Pass if arov is set
                                        STOP^
033454  254 04 0 00 033455              HALT    .+1                     ;Test failed if program halts here
033455  324 00 0 00 033456              JUMPA   .+1                     ;To loop change this instruction^
033456  255 02 0 00 033460              JCRY1   .+2                     ;Pass if cry0 is set
                                        STOP^
033457  254 04 0 00 033460              HALT    .+1                     ;Test failed if program halts here
033460  324 00 0 00 033461              JUMPA   .+1                     ;To loop change this instruction^
033461  255 04 0 00 033463              JCRY0   .+2                     ;Pass if cry0 is reset
033462  334 00 0 00 000000              SKIPA
                                        STOP^
033463  254 04 0 00 033464              HALT    .+1                     ;Test failed if program halts here
033464  324 00 0 00 033465              JUMPA   .+1                     ;To loop change this instruction^
033465  255 01 0 00 033467              JFOV    .+2                     ;Pass if fov is reset
033466  334 00 0 00 000000              SKIPA
                                        STOP^
033467  254 04 0 00 033470              HALT    .+1                     ;Test failed if program halts here
033470  324 00 0 00 033471              JUMPA   .+1                     ;To loop change tnis instruction^

                                ;**********
```

```
5272                                    ;This test verifies that movm will set the arov and
5273                                    ;Cry1 flags only when the data is 400000,,0
5274                                    ;First, the flags are reset; then, movm [400000,,0] is executed.
5275                                    ;The arithmetic flags are checked.
5276                                    ;If arov and cry1 are set and cry0 and fov are reset, this test passes.
5277
5278  033471  255 17 0 00 033472        C12301: JFCL    17,.+1                  ;Clear arithmetic flags
5279  033472  214 00 0 00 034442                MOVM    [XWD 400000,0]          ;*Movm [400000,,0] should set arov and cry1 only
5280  033473  255 10 0 00 033475                JOV     .+2                     ;Pass if arov is set
5281                                            STOP^
5282  033474  254 04 0 00 033475                HALT    .+1                     ;Test failed if program halts here
5283  033475  324 00 0 00 033476                JUMPA   .+1                     ;To loop change this instruction^
5284  033476  255 02 0 00 033500                JCRY1   .+2                     ;Pass if cry0 is set
5285                                            STOP^
5286  033477  254 04 0 00 033500                HALT    .+1                     ;Test failed if program halts here
5287  033500  324 00 0 00 033501                JUMPA   .+1                     ;To loop change this instruction^
5288  033501  255 04 0 00 033503                JCRY0   .+2                     ;Pass if cry0 is reset
5289  033502  334 00 0 00 000000                SKIPA
5290                                            STOP^
5291  033503  254 04 0 00 033504                HALT    .+1                     ;Test failed if program halts here
5292  033504  324 00 0 00 033505                JUMPA   .+1                     ;To loop change this instruction^
5293  033505  255 01 0 00 033507                JFOV    .+2                     ;Pass if fov is reset
5294  033506  334 00 0 00 000000                SKIPA
5295                                            STOP^
5296  033507  254 04 0 00 033510                HALT    .+1                     ;Test failed if program halts here
5297  033510  324 00 0 00 033511                JUMPA   .+1                     ;To loop change this instruction^
5298
5299                                  ;**********
```

```
5300                                    SUBTTL  TEST OF AOS AND SOS INSTRUCTIONS
5301
5302                                    ;**********
5303
5304                                    ;This test verifies that aos adds 1 to memory and does not skip.
5305                                    ;First, e is cleared; then, aos is executed. next e is checked for 1.
5306                                    ;If c(e) is not 1 or aos skipped, this test fails.
5307
5308  033511  400 00 0 00 000000        C12600: SETZ                            ;Clear e
5309  033512  350 00 0 00 000000                AOS                           ;*Aos should add 1 to c(e) and not skip
5310  033513  302 00 0 00 000001                CAIE    1                     ;Pass if c(e)=0,,1 andaos did not skip
5311                                            STOP^
5312  033514  254 04 0 00 033515                HALT    .+1                   ;Test failed if program halts here
5313  033515  324 00 0 00 033516                JUMPA   .+1                   ;To loop change this instruction^
5314
5315                                    ;**********
5316
5317                                    ;This test verifies that aos adds 1 to memory and does not skip.  first,
5318                                    ;E is preloaded with -1,,-1; then, aos is executed. next e is checked for 0.
5319                                    ;If c(e) is not 0 or aos skipped, this test fails.
5320
5321  033516  474 00 0 00 000000        C12700: SETO                            ;Preload e with -1,,-1
5322  033517  350 00 0 00 000000                AOS                           ;*Aos should add 1 to c(e) and not skip
5323  033520  302 00 0 00 000000                CAIE                          ;Pass if c(e)=0,,0 and aos did not skip
5324                                            STOP^
5325  033521  254 04 0 00 033522                HALT    .+1                   ;Test failed if program halts here
5326  033522  324 00 0 00 033523                JUMPA   .+1                   ;To loop change this instruction^
5327
5328                                    ;**********
```

```
5329                                    ;This test verifies that sos subtracts 1 from memory and does not skip.
5330                                    ;First, e is cleared; then, sos is executed. next e is checked for -1,,-1.
5331                                    ;If c(e) is not -1,,-1 or sos skipped, this test fails.
5332
5333  033523  400 00 0 00 000000       C13100: SETZ                            ;Clear e
5334  033524  370 00 0 00 000000               SOS                             ;*Sos should subtract 1 from c(e) and not skip
5335  033525  312 00 0 00 034443               CAME    [-1]                    ;Pass if c(e)=-1,,-1 and sos did not skip
5336                                            STOP^
5337  033526  254 04 0 00 033527               HALT    .+1                     ;Test failed if program halts here
5338  033527  324 00 0 00 033530               JUMPA   .+1                     ;To loop change this instruction^
5339
5340                                    ;**********
5341
5342                                    ;This test verifies that sos subtracts 1 from memory and does not skip.
5343                                    ;First,   is preloaded with a; then, sos is executed. next e is checked for 0.
5344                                    ;If c(e) is not 0 or sos skipped, this test fails.
5345
5346  033530  201 00 0 00 000001       C13200: MOVEI   1                       ;Preload e with 1
5347  033531  370 00 0 00 000000               SOS                             ;*Sos should subtract 1 from c(e) and did not skip
5348  033532  302 00 0 00 000000               CAIE    0                       ;Pass if c(e)=0 and sos did not skip
5349                                            STOP^
5350  033533  254 04 0 00 033534               HALT    .+1                     ;Test failed if program halts here
5351  033534  324 00 0 00 033535               JUMPA   .+1                     ;To loop change this instruction^
5352
5353                                    ;**********
```

```
5354                                    ;This test verifies that sos sets cry0/1 flags if c(e) is a non-zero number
5355                                    ;Other than 400000,,0.  in this case, c(e)=1.  first the flags are reset; then
5356                                    ;Sos is executed.  the  flags are checked.  if cry0 and cry1 are set and arov
5357                                    ;And fov are reset, this test passes.
5358
5359  033535  255 17 0 00 033536        C13300: JFCL    17,.+1                  ;Reset arithmetic flags
5360  033536  201 00 0 00 000001                MOVEI   1                       ;Preload e with 1
5361  033537  370 00 0 00 000000                SOS                             ;*Sos should set cry0/1
5362  033540  255 04 0 00 033542                JCRY0   .+2                     ;Pass if cry0 is set
5363                                            STOP^
5364  033541  254 04 0 00 033542                HALT    .+1                     ;Test failed if program halts here
5365  033542  324 00 0 00 033543                JUMPA   .+1                     ;To loop change this instruction^
5366  033543  255 02 0 00 033545                JCRY1   .+2                     ;Pass if cry1 is set
5367                                            STOP^
5368  033544  254 04 0 00 033545                HALT    .+1                     ;Test failed if program halts here
5369  033545  324 00 0 00 033546                JUMPA   .+1                     ;To loop change this instruction^
5370  033546  255 10 0 00 033550                JOV     .+2                     ;Pass if arov is reset
5371  033547  334 00 0 00 000000                SKIPA
5372                                            STOP^
5373  033550  254 04 0 00 033551                HALT    .+1                     ;Test failed if program halts here
5374  033551  324 00 0 00 033552                JUMPA   .+1                     ;To loop change this instruction^
5375  033552  255 01 0 00 033554                JFOV    .+2                     ;Pass if fov is reset
5376  033553  334 00 0 00 000000                SKIPA
5377                                            STOP^
5378  033554  254 04 0 00 033555                HALT    .+1                     ;Test failed if program halts here
5379  033555  324 00 0 00 033556                JUMPA   .+1                     ;To loop change tnis instruction^
5380
5381                                    ;**********
```

```
                                ;This test verifies that aos sets cry0 and cry1 flag if c(e) is -1,,-1
                                ;First the flags are reset; then aos is executed.  the flags are checked.
                                ;If cry0 and cry1 are set and arov and fov are reset, this test passes.

033556  255 17 0 00 033557      C13400: JFCL    17,.+1                  ;Reset arithmetic flags
033557  474 00 0 00 000000              SETO                            ;Preload e with 1
033560  350 00 0 00 000000              AOS                             ;*Aos should set cry0 and cry1
033561  255 04 0 00 033563              JCRY0   .+2                     ;Pass if cry0 is set
                                        STOP^
033562  254 04 0 00 033563              HALT    .+1                     ;Test failed if program halts here
033563  324 00 0 00 033564              JUMPA   .+1                     ;To loop change this instruction^
033564  255 02 0 00 033566              JCRY1   .+2                     ;Pass if cry1 is set
                                        STOP^
033565  254 04 0 00 033566              HALT    .+1                     ;Test failed if program halts here
033566  324 00 0 00 033567              JUMPA   .+1                     ;To loop change this instruction^
033567  255 10 0 00 033571              JOV     .+2                     ;Pass if arov is reset
033570  334 00 0 00 000000              SKIPA
                                        STOP^
033571  254 04 0 00 033572              HALT    .+1                     ;Test failed if program halts here
033572  324 00 0 00 033573              JUMPA   .+1                     ;To loop change this instruction^
033573  255 01 0 00 033575              JFOV    .+2                     ;Pass if fov is reset
033574  334 00 0 00 000000              SKIPA
                                        STOP^
033575  254 04 0 00 033576              HALT    .+1                     ;Test failed if program halts here
033576  324 00 0 00 033577              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
                                        SUBTTL  TEST OF INTERACTION OF JFCL, JRST, AND JSP WITH ARITHMETIC FLAGS

                                        ;**********

                                        ;This test verifies that jfcl 1, will always clear fov flag.
                                        ;First, fov is set via jrst 2, ;then jfcl 1,.+1 is executed to clear fov.
                                        ;If fov was cleared, this test passes.

033577                                  C13600: SFLAG   FOV                     ^;Set fov flag

033577  205 01 0 00 040000                      MOVSI   1,FOV
033600  255 17 0 00 033601                      JFCL    17,.+1                  ;Reset all flags
033601  254 02 0 01 033602                      JRST    2,.+1(1)                ;Set FOV flag
033602  255 01 0 00 033603                      JFCL    1,.+1                   ;*Jfcl should reset fov
033603  255 01 0 00 033605                      JFCL    1,.+2                   ;Pass if fov is reset
033604  334 00 0 00 000000                      SKIPA                           ;Skip halt if jfcl 1,.+1 passed
                                                STOP^
033605  254 04 0 00 033606                      HALT    .+1                     ;Test failed if program halts here
033606  324 00 0 00 033607                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that jrst 2, can set fxu and jsp can save fxu in the ac.
                                        ;First, fxu is set via jrst 2, ;then, jsp is executed. the ac is checked for
                                        ;Fxu. if fxu is set, this test passes; otherwise, either jrst 2, or jsp failed.

033607                                  C13700: SFLAG   FXU                     ^;*Set fxu flag

033607  205 01 0 00 000100                      MOVSI   1,FXU
033610  255 17 0 00 033611                      JFCL    17,.+1                  ;Reset all flags
033611  254 02 0 01 033612                      JRST    2,.+1(1)                ;Set FXU flag
033612  265 00 0 00 033613                      JSP     .+1                     ;*Store fxu flag in ac
033613  607 00 0 00 000100                      TLNN    FXU                     ;Pass if fxu is set in the ac
                                                STOP^
033614  254 04 0 00 033615                      HALT    .+1                     ;Test failed if program halts here
033615  324 00 0 00 033616                      JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
5447                                    ;This test verifies that jrst 2, can reset fxu and jsp can save fxu in the ac.
5448                                    ;First, fxu is set; then, fxu is reset via jrst 2,.  next, jsp is executed; and
5449                                    ;The ac is checked for fxu reset.  if fxu is reset in the ac, this test passes;
5450                                    ;Otherwise, jrst 2, failed to clear fxu or jsp stored fxu incorectly.
5451
5452  033616                            C14000: SFLAG   FXU                     ^;Set fxu flag
5453
5454  033616  205 01 0 00 000100                MOVSI   1,FXU
5455  033617  255 17 0 00 033620                JFCL    17,.+1                  ;Reset all flags
5456  033620  254 02 0 01 033621                JRST    2,.+1(1)                ;Set FXU flag
5457                                            SFLAG                           ^;*Reset fxu flag
5458
5459  033621  205 01 0 00 000000                MOVSI   1,
5460  033622  255 17 0 00 033623                JFCL    17,.+1                  ;Reset all flags
5461  033623  254 02 0 01 033624                JRST    2,.+1(1)                ;Set  flag
5462  033624  265 00 0 00 033625                JSP     .+1                     ;*Store fxu flag in the ac
5463  033625  603 00 0 00 000100                TLNE    FXU                     ;Pass if fxu is reset in the ac
5464                                            STOP^
5465  033626  254 04 0 00 033627                HALT    .+1                     ;Test failed if program halts here
5466  033627  324 00 0 00 033630                JUMPA   .+1                     ;To loop change this instruction^
5467
5468                                    ;**********
5469
5470                                    ;This test verifies that jrst 2, can set dck and jsp can save dck in the ac.
5471                                    ;First, dck is set via jrst 2, ;then jsp is executed.  the ac is checked
5472                                    ;For dck.  if dck is set, this test passes, otherwise jrst 2, or jsp failed.
5473
5474  033630                            C14100: SF_AG   DCK                     ^;*Set dck flag
5475
5476  033630  205 01 0 00 000040                MOVSI   1,DCK
5477  033631  255 17 0 00 033632                JFCL    17,.+1                  ;Reset all flags
5478  033632  254 02 0 01 033633                JRST    2,.+1(1)                ;Set DCK flag
5479  033633  265 00 0 00 033634                JSP     .+1                     ;*Store fxu flag in ac
5480  033634  607 00 0 00 000040                TLNN    DCK                     ;Pass if fxu is set in the ac
5481                                            STOP^
5482  033635  254 04 0 00 033636                HALT    .+1                     ;Test failed if program halts here
5483  033636  324 00 0 00 033637                JUMPA   .+1                     ;To loop change this instruction^
5484
5485                                    ;**********
```

```
5486                                    ;This test verifies that jrst 2, can reset dck and jsp can save dck in the ac.
5487                                    ;First, fxu is set; then, dck is reset via jrst 2,.  next, jsp is expected; and
5488                                    ;The ac is checked for dck reset.  if dck is reset in the ac, this test passes;
5489                                    ;Otherwise, jrst 2, failed to clear dck or jsp stored dck incorrectly.
5490
5491  033637                            C14200: SFLAG   DCK                     ^;Set dck flag
5492
5493  033637  205 01 0 00 000040                MOVSI   1,DCK
5494  033640  255 17 0 00 033641                JFCL    17,.+1                ;Reset all flags
5495  033641  254 02 0 01 033642                JRST    2,.+1(1)              ;Set DCK flag
5496                                            SFLAG                         ^;*Reset dck flag
5497
5498  033642  205 01 0 00 000000                MOVSI   1,
5499  033643  255 17 0 00 033644                JFCL    17,.+1                ;Reset all flags
5500  033644  254 02 0 01 033645                JRST    2,.+1(1)              ;Set  flag
5501  033645  265 00 0 00 033646                JSP     .+1                   ;*Store dck flag in the ac
5502  033646  603 00 0 00 000040                TLNE    DCK                   ;Pass if dck is reset in the ac
5503                                            STOP^
5504  033647  254 04 0 00 033650                HALT    .+1                   ;Test failed if program halts here
5505  033650  324 00 0 00 033651                JUMPA   .+1                   ;To loop change this instruction^
5506
5507                                    ;**********
```

```
                                SUBTTL  TEST OF JUMPX INSTRUCTIONS

                                ;**********

                                ;This test verifies that jumpl is data dependent.  it will jump if and only
                                ;If c(ac) is negative.  in this test, the ac contains -1,,-1.  hence, jumpl
                                ;Should jump.  if jumpl jumps, this test passes.

033651  474 00 0 00 000000       C14500: SETO                            ;Preload ac with -1,,-1
033652  321 00 0 00 033654               JUMPL   .+2                     ;*Jumpl should jump because c(ac) is negative
                                         STOP^
033653  254 04 0 00 033654               HALT    .+1                     ;Test failed if program halts here
033654  324 00 0 00 033655               JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that jumpl is data dependent.  it will jump if and only
                                ;If c(ac) is negative.  in this test, the ac contains 0.  hence, jumpl
                                ;Should not jump.  if jumpl does not jump, this test passes.

033655  400 00 0 00 000000       C14600: SETZ                            ;Preload ac with 0
033656  321 00 0 00 033660               JUMPL   .+2                     ;*Jumpl should not jump
033657  334 00 0 00 000000               SKIPA                           ;Pass if jumpl does not jump
                                         STOP^
033660  254 04 0 00 033661               HALT    .+1                     ;Test failed if program halts here
033661  324 00 0 00 033662               JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that jumpe is data dependent.  it will jump if and
                                ;Only if c(ac)=0.  in this test, the ac contains 0. hence, jumpe should
                                ;Jump.  if jumpe jumps, this test passes.

033662  400 00 0 00 000000       C14700: SETZ                            ;Preload ac with 0
033663  322 00 0 00 033665               JUMPE   .+2                     ;*Jumpe should jump
                                         STOP^
033664  254 04 0 00 033665               HALT    .+1                     ;Test failed if program halts here
033665  324 00 0 00 033666               JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
5548                                    ;This test verifies that jumpn is data dependent.  it will jump if and
5549                                    ;Only if c(ac) is non-zero.  in this test, the ac contains 1.  hence,
5550                                    ;Jumpn should jump.  if jumpn jumps, this test passes.
5551
5552  033666  201 00 0 00 000001        C15000: MOVEI   1                       ;Preload ac with 1
5553  033667  326 00 0 00 033671                JUMPN   .+2                     ;*Jumpn should jump
5554                                            STOP^
5555  033670  254 04 0 00 033671                HALT    .+1                     ;Test failed if program halts here
5556  033671  324 00 0 00 033672                JUMPA   .+1                     ;To loop change this instruction^
5557
5558                                    ;**********
5559
5560                                    ;This test verifies that jumpe is data dependent.  it will jump if and
5561                                    ;Only if c(ac)=0.  in this test, the ac contains 2.  hence, jumpl should
5562                                    ;Not jump.  if jumpe does not jump, this test passes.
5563
5564  033672  201 00 0 00 000002        C15100: MOVEI   2                       ;Preload ac with 2
5565  033673  322 00 0 00 033675                JUMPE   .+2                     ;*Jumpe should not jump
5566  033674  334 00 0 00 000000                SKIPA                           ;Pass if jumpe did not jump
5567                                            STOP^
5568  033675  254 04 0 00 033676                HALT    .+1                     ;Test failed if program halts here
5569  033676  324 00 0 00 033677                JUMPA   .+1                     ;To loop change this instruction^
5570
5571                                    ;**********
5572
5573                                    ;This test verifies that jumpn is data dependent.  it will jump if and
5574                                    ;Only if c(ac) is non-zero.  in this test, the ac contains 0.  hence,
5575                                    ;Jumpn should not jump.  if jumpn does not jump, this test passes.
5576
5577  033677  400 00 0 00 000000        C15200: SETZ                            ;Preload ac with 0
5578  033700  326 00 0 00 033702                JUMPN   .+2                     ;*Jumpn should not jump
5579  033701  334 00 0 00 000000                SKIPA                           ;Pass if jumpn did not jump
5580                                            STOP^
5581  033702  254 04 0 00 033703                HALT    .+1                     ;Test failed if program halts here
5582  033703  324 00 0 00 033704                JUMPA   .+1                     ;To loop change this instruction^
5583
5584                                    ;**********
```

```
 5585                                   ;This test verifies that jumpg is data dependent.  it will jump if and
 5586                                   ;Only if c(ac) is greater than 0.  in this test, the ac contains 1.  hence,
 5587                                   ;Jumpg should jump.  if jumpg jumps, this test passes.
 5588
 5589  033704  201 00 0 00 000001       C15300: MOVEI   1                       ;Preload  ac with 1
 5590  033705  327 00 0 00 033707               JUMPG   .+2                     ;*Jumpg should jump
 5591                                           STOP^
 5592  033706  254 04 0 00 033707               HALT    .+1                     ;Test failed if program halts here
 5593  033707  324 00 0 00 033710               JUMPA   .+1                     ;To loop change this instruction^
 5594
 5595                                   ;**********
 5596
 5597                                   ;This test verifies that jumpg is data dependent.  it will jump if and only
 5598                                   ;If c(ac) is greater than 0.  in this test, the ac contains -1,,0.  hence,
 5599                                   ;Jumpg should not jump.  if jumpg does not jump, this test passes.
 5600
 5601  033710  205 00 0 00 777777       C15400: MOVSI   -1                      ;Preload ac with -1,,0
 5602  033711  327 00 0 00 033713               JUMPG   .+2                     ;*Jumpg should not jump
 5603  033712  334 00 0 00 000000               SKIPA                           ;Pass if jumpg did not jump
 5604                                           STOP^
 5605  033713  254 04 0 00 033714               HALT    .+1                     ;Test failed if program halts here
 5606  033714  324 00 0 00 033715               JUMPA   .+1                     ;To loop change this instruction^
 5607
 5608                                   ;**********
```

```
                                SUBTTL  TEST OF AOJ AND SOJ INSTRUCTIONS

                                ;**********

                                ;This test verifies that aoj adds 1 to the ac and does not jump.
                                ;First, the ac is preloaded with 0; then, aoj is executed.  next, the
                                ;Ac is checked for 1.  if c(ac) is not 1 or aoj skipped, this test fails.

033715  400 00 0 00 000000      C15500: SETZ                            ;Preload ac with 0
033716  340 00 0 00 033720              AOJ     .+2                     ;*Aoj should add 1 to the ac and not jump
033717  302 00 0 00 000001              CAIE    1                       ;Pass if c(ac)=1 and aoj did not jump
                                        STOP^
033720  254 04 0 00 033721              HALT    .+1                     ;Test failed if program halts here
033721  324 00 0 00 033722              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that aoj adds 1 to the ac and does not jump.
                                ;First, the ac is preloaded with -1,,-1; then, aoj is executed.  next, the
                                ;Ac is checked for 0.  if c(ac) is not 0 or aoj skipped, this test fails.

033722  474 00 0 00 000000      C15600: SETO                            ;Preload ac with 0
033723  340 00 0 00 033725              AOJ     .+2                     ;*Aoj should add 1 to the ac and not jump
033724  302 00 0 00 000000              CAIE    0                       ;Pass if c(ac)=1 and aoj did not jump
                                        STOP^
033725  254 04 0 00 033726              HALT    .+1                     ;Test failed if program halts here
033726  324 00 0 00 033727              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that soj subtracts 1 from the ac and does not jump.
                                ;First, the ac is preloaded with 0; then, soj is executed.  next, the ac
                                ;Is checked for -1,,-1. if c(ac) is not -1,,-1 or soj skipped, this test fails.

033727  400 00 0 00 000000      C15700: SETZ                            ;Preload ac with 0
033730  360 00 0 00 033732              SOJ     .+2                     ;*Soj should subtract 1 from the ac and not jump
033731  312 00 0 00 034443              CAME    [-1]                    ;Pass if c(ac)=-1,,-1 and soj did not jump
                                        STOP^
033732  254 04 0 00 033733              HALT    .+1                     ;Test failed if program halts here
033733  324 00 0 00 033734              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
5651                                    ;This test verifies that soj subtracts 1 from the ac and does not jump.
5652                                    ;First, the ac is preloaded with -1,,-1; then, soj is executed.  next, the ac
5653                                    ;Is checked for -1,,-2. if c(ac) is not -1,,-2 or soj skipped, this test fails.
5654
5655  033734  474 00 0 00 000000       C16000: SETO                            ;Preload ac with -1,,-1
5656  033735  360 00 0 00 033737               SOJ     .+2                     ;*Soj should subtract 1 from the ac and not jump
5657  033736  312 00 0 00 034444               CAME    [-2]                    ;Pass if c(ac)=-1,,-2 and soj did not jump
5658                                           STOP^
5659  033737  254 04 0 00 033740               HALT    .+1                     ;Test failed if program halts here
5660  033740  324 00 0 00 033741               JUMPA   .+1                     ;To loop change this instruction^
5661
5662                                    ;**********
5663
5664                                    ;This test verifies that soj subtracts 1 from the ac and does not jump.
5665                                    ;First, the ac is preloaded with 0,,1; then, soj is executed.  next, the
5666                                    ;Ac is checked for 0.  if c(ac) is not 0 or soj skipped, this test fails.
5667
5668  033741  201 00 0 00 000001       C16100: MOVEI   1                       ;Preload ac with 1
5669  033742  360 00 0 00 033744               SOJ     .+2                     ;*Soj should subtract 1 from the ac and not jump
5670  033743  302 00 0 00 000000               CAIE    0                       ;Pass if c(ac)=0 and soj did not jump
5671                                           STOP^
5672  033744  254 04 0 00 033745               HALT    .+1                     ;Test failed if program halts here
5673  033745  324 00 0 00 033746               JUMPA   .+1                     ;To loop change this instruction^
5674
5675                                    ;**********
```

```
                                ;This test verifies that soj ac, followed by aoj ac, has no net effect on c(ac).
                                ;In this case, the ac is preloaded with 0; then, soj ac, followed by aoj.
                                ;Ac, is repeated 7 times.  the ac is then checked for its original contents, 0.
                                ;If c(ac)=0, this test passes; otherwise aoj or soj failed.

                     000017     C16200: AC=17
033746  400 17 0 00 000000              SETZ    AC,                     ;Preload ac with 0
                                        REPEAT  ^D10,<
                                        SOJ     AC,.                    ;*Soj should subtract 1 from the ac
                                        AOJ     AC,.                    ;*Aoj should add 1 to the ac>

033747  360 17 0 00 033747              SOJ     AC,.                    ;*Soj should subtract 1 from the ac
033750  340 17 0 00 033750              AOJ     AC,.                    ;*Aoj should add 1 to the ac

033751  360 17 0 00 033751              SOJ     AC,.                    ;*Soj should subtract 1 from the ac
033752  340 17 0 00 033752              AOJ     AC,.                    ;*Aoj should add 1 to the ac

033753  360 17 0 00 033753              SOJ     AC,.                    ;*Soj should subtract 1 from the ac
033754  340 17 0 00 033754              AOJ     AC,.                    ;*Aoj should add 1 to the ac

033755  360 17 0 00 033755              SOJ     AC,.                    ;*Soj should subtract 1 from the ac
033756  340 17 0 00 033756              AOJ     AC,.                    ;*Aoj should add 1 to the ac

033757  360 17 0 00 033757              SOJ     AC,.                    ;*Soj should subtract 1 from the ac
033760  340 17 0 00 033760              AOJ     AC,.                    ;*Aoj should add 1 to the ac

033761  360 17 0 00 033761              SOJ     AC,.                    ;*Soj should subtract 1 from the ac
033762  340 17 0 00 033762              AOJ     AC,.                    ;*Aoj should add 1 to the ac

033763  360 17 0 00 033763              SOJ     AC,.                    ;*Soj should subtract 1 from the ac
033764  340 17 0 00 033764              AOJ     AC,.                    ;*Aoj should add 1 to the ac

033765  360 17 0 00 033765              SOJ     AC,.                    ;*Soj should subtract 1 from the ac
033766  340 17 0 00 033766              AOJ     AC,.                    ;*Aoj should add 1 to the ac

033767  360 17 0 00 033767              SOJ     AC,.                    ;*Soj should subtract 1 from the ac
033770  340 17 0 00 033770              AOJ     AC,.                    ;*Aoj should add 1 to the ac

033771  360 17 0 00 033771              SOJ     AC,.                    ;*Soj should subtract 1 from the ac
033772  340 17 0 00 033772              AOJ     AC,.                    ;*Aoj should add 1 to the ac
033773  332 00 0 00 000017              SKIPE   AC                      ;Pass if c(ac) is unchanged. i.e. c(ac)=0
                                        STOP^
033774  254 04 0 00 033775              HALT    .+1                     ;Test failed if program halts here
033775  324 00 0 00 033776              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
5722                                    ;This test verifies that soj sets cry0/1 flags if c(e) is a non-zero number
5723                                    ;Other  than  400000,,0.   in  this case, c(e)=1.  first the flags are reset.
5724                                    ;Then soj is executed.  the  flags are checked.  if cry0 and cry1 are set and
5725                                    ;Arov and fov are reset, this test passes.
5726
5727  033776  255 17 0 00 033777        C16201: JFCL    17,..+1                 ;Reset arithmetic flags
5728  033777  201 00 0 00 000001                MOVEI   1                       ;Preload e with 1
5729  034000  360 00 0 00 000000                SOJ                             ;*Soj should set cry0/1
5730  034001  255 04 0 00 034003                JCRY0   .+2                     ;Pass if cry0 is set
5731                                            STOP^
5732  034002  254 04 0 00 034003                HALT    .+1                     ;Test failed if program halts here
5733  034003  324 00 0 00 034004                JUMPA   .+1                     ;To loop change this instruction^
5734  034004  255 02 0 00 034006                JCRY1   .+2                     ;Pass if cry1 is set
5735                                            STOP^
5736  034005  254 04 0 00 034006                HALT    .+1                     ;Test failed if program halts here
5737  034006  324 00 0 00 034007                JUMPA   .+1                     ;To loop change this instruction^
5738  034007  255 10 0 00 034011                JOV     .+2                     ;Pass if arov is reset
5739  034010  334 00 0 00 000000                SKIPA
5740                                            STOP^
5741  034011  254 04 0 00 034012                HALT    .+1                     ;Test failed if program halts here
5742  034012  324 00 0 00 034013                JUMPA   .+1                     ;To loop change this instruction^
5743  034013  255 01 0 00 034015                JFOV    .+2                     ;Pass if fov is reset
5744  034014  334 00 0 00 000000                SKIPA
5745                                            STOP^
5746  034015  254 04 0 00 034016                HALT    .+1                     ;Test failed if program halts here
5747  034016  324 00 0 00 034017                JUMPA   .+1                     ;To loop change this instruction^
5748
5749                                    ;**********
```

```
5750                                    ;This test verifies that aoj sets cry0 and cry1 flag if c(e) is -1,,-1.
5751                                    ;First the flags are reset; then aoj is executed.  the flags are checked.
5752                                    ;If cry0 and cry1 are set and arov and fov are reset, this test passes.
5753
5754  034017  255 17 0 00 034020       C16202: JFCL    17,.+1                  ;Reset arithmetic flags
5755  034020  474 00 0 00 000000               SETO                            ;Preload e with 1
5756  034021  340 00 0 00 000000               AOJ                             ;*Aoj should set cry0 and cry1
5757  034022  255 04 0 00 034024               JCRY0   .+2                     ;Pass if cry0 is set
5758                                           STOP^
5759  034023  254 04 0 00 034024               HALT    .+1                     ;Test failed if program halts here
5760  034024  324 00 0 00 034025               JUMPA   .+1                     ;To loop change this instruction^
5761  034025  255 02 0 00 034027               JCRY1   .+2                     ;Pass if cry1 is set
5762                                           STOP^
5763  034026  254 04 0 00 034027               HALT    .+1                     ;Test failed if program halts here
5764  034027  324 00 0 00 034030               JUMPA   .+1                     ;To loop change this instruction^
5765  034030  255 10 0 00 034032               JOV     .+2                     ;Pass if arov is reset
5766  034031  334 00 0 00 000000               SKIPA
5767                                           STOP^
5768  034032  254 04 0 00 034033               HALT    .+1                     ;Test failed if program halts here
5769  034033  324 00 0 00 034034               JUMPA   .+1                     ;To loop change this instruction^
5770  034034  255 01 0 00 034036               JFOV    .+2                     ;Pass if fov is reset
5771  034035  334 00 0 00 000000               SKIPA
5772                                           STOP^
5773  034036  254 04 0 00 034037               HALT    .+1                     ;Test failed if program halts here
5774  034037  324 00 0 00 034040               JUMPA   .+1                     ;To loop change this instruction^
5775
5776                                    ;**********
```

```
                                        SUBTTL  TEST OF MEMORY, BOTH AND SELF MODE INSTRUCTIONS

                                        ;**********

                                        ;This test verifies that addm does not modify c(ac)
                                        ;Both ac and e are preloaded with -1,,-1.  then addm is executed.
                                        ;This test passes if c(ac) are unchanged.

034040  477 00 0 00 000001       C16400: SETOB   1                       ;Preload ac, e with -1,,-1
034041  272 00 0 00 000001               ADDM    1                       ;*Addm should not affect c(ac)
034042  312 00 0 00 034443               CAME    [-1]                    ;Pass if c(ac) is unmodified by addm
                                        STOP^
034043  254 04 0 00 034044               HALT    .+1                     ;Test failed if program halts here
034044  324 00 0 00 034045               JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that hrrem does not modify c(ac)
                                        ;The ac is preloaded with 0,,-1.  then hrrm is executed.
                                        ;This test passes if c(ac) are unchanged.

034045  201 00 0 00 777777       C16500: MOVEI   -1                      ;Preload ac with 0,,-1
034046  572 00 0 00 000001               HRREM   1                       ;*Hrrm should not affect c(ac)
034047  302 00 0 00 777777               CAIE    -1                      ;Pass if c(ac) is unmodified by hrrm
                                        STOP^
034050  254 04 0 00 034051               HALT    .+1                     ;Test failed if program halts here
034051  324 00 0 00 034052               JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                        ;This test verifies that movsm does not modify c(ac).
                                        ;The ac is preloaded with 0,,-1.  then movsm is executed.
                                        ;This test passes if c(ac) are unchanged.

034052  201 00 0 00 777777      C16600: MOVEI   -1                      ;Preload ac with 0,,-1
034053  206 00 0 00 000001              MOVSM   1                       ;*Movsm should not affect c(ac)
034054  302 00 0 00 777777              CAIE    -1                      ;Pass if c(ac) is unmodified by movsm
                                        STOP^
034055  254 04 0 00 034056              HALT    .+1                     ;Test failed if program halts here
034056  324 00 0 00 034057              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that xorm does not modify c(ac).
                                        ;Both ac and e are preloaded with -1,,-1.  then xorm is executed.
                                        ;This test passes if c(ac) are unchanged.

034057  477 00 0 00 000001      C16700: SETOB   1                       ;Preload ac, e with -1,,-1
034060  432 00 0 00 000001              XORM    1                       ;*Xorm should not affect c(ac)
034061  312 00 0 00 034443              CAME    [-1]                    ;Pass if c(ac) is unmodified by xorm
                                        STOP^
034062  254 04 0 00 034063              HALT    .+1                     ;Test failed if program halts here
034063  324 00 0 00 034064              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
5831                                    ;This test verifies that addb adds c(ac) to c(e) and places the result
5832                                    ;In both ac and e. in this test, both ac and e are preloaded with -1,,-1.
5833                                    ;Then, addb is executed.  c(ac) is  then  compared  to c(e) and c(ac) is
5834                                    ;Compared to -2.  if both of these comparisons succeed, this test passes.
5835                                    ;Otherwise, addb failed.
5836
5837  034064  477 00 0 00 000001       C17000: SETOB   1                       ;Preload ac, e with -1,,-1
5838  034065  273 00 0 00 000001               ADDB    1                       ;*Addb should add c(ac) to c(e) and place result
5839                                                                          ;Into both ac and e
5840  034066  312 00 0 00 000001               CAME    1                       ;Pass if c(ac)=c(e)
5841                                            STOP^
5842  034067  254 04 0 00 034070               HALT    .+1                     ;Test failed if program halts here
5843  034070  324 00 0 00 034071               JUMPA   .+1                     ;To loop change this instruction^
5844  034071  312 00 0 00 034444               CAME    [-2]                    ;Pass if c(ac)=-2
5845                                            STOP^
5846  034072  254 04 0 00 034073               HALT    .+1                     ;Test failed if program halts here
5847  034073  324 00 0 00 034074               JUMPA   .+1                     ;To loop change this instruction^
5848
5849                                    ;*********
5850
5851                                    ;This test verifies that addm adds c(ac) to c(e) and places the result in e
5852                                    ;In this case, ac, e are both preloaded with -1; then, addm is executed. e is
5853                                    ;Then checked for -2.  if c(e)=-2, this test passes; otherwise, addm failed.
5854
5855  034074  477 00 0 00 000001       C17100: SETOB   1                       ;Preload ac, e with -1,,-1
5856  034075  272 00 0 00 000001               ADDM    1                       ;*Addm should add c(ac) to c(e)
5857  034076  312 01 0 00 034444               CAME    1,[-2]                  ;Pass if c(e)=-2
5858                                            STOP^
5859  034077  254 04 0 00 034100               HALT    .+1                     ;Test failed if program halts here
5860  034100  324 00 0 00 034101               JUMPA   .+1                     ;To loop change this instruction^
5861
5862                                    ;**********
```

```
                                        ;This test verifies that hllos places ones on the right half of e
                                        ;But does not affect the left half of e.  in this case,
                                        ;E is preloaded with 0; the, hllos is executed.  the result
                                        ;In e should be 0,,-1.  if c(e)=0,,-1, this test passes.

034101  402 00 0 00 000001      C17200: SETZM   1                       ;Preload e with 0
034102  523 00 0 00 000001              HLLOS   1                       ;*Hllos should place 0,,-1 into e
034103  302 01 0 00 777777              CAIE    1,-1                    ;Pass if c(e)=0,,-1
                                        STOP^
034104  254 04 0 00 034105              HALT    .+1                     ;Test failed if program halts here
034105  324 00 0 00 034106              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that movss swaps boths halves of e and
                                        ;Places the result in e.  in this case, e is preloaded with
                                        ;-1,,0; Then, movss is executed.  the result in e should be 0,,-1.
                                        ;If c(e)=0,,-1, this test passes.

034106  205 01 0 00 777777      C17300: MOVSI   1,-1                    ;Preload e with -1,,0
034107  207 00 0 00 000001              MOVSS   1                       ;*Movss should place 0,,-1 into e
034110  302 01 0 00 777777              CAIE    1,-1                    ;Pass if c(e)=0,,-1
                                        STOP^
034111  254 04 0 00 034112              HALT    .+1                     ;Test failed if program halts here
034112  324 00 0 00 034113              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                ;This test verifies that aos adds one to memory but does not skip.  first,
                                ;E is preloaded with -1,-1; then, aos is executed.  next, e is checked for 0.
                                ;If c(e) is not 0 or aos skipped, this test fails.

034113  477 00 0 00 000001      C17400: SETOB   1                       ;Preload e with -1,,-1
034114  350 00 0 00 000001              AOS     1                       ;*Aos should add to c(e) and not skip
034115  302 01 0 00 000000              CAIE    1,0                     ;Pass if c(e)=0,,0 and aos did not skip
                                        STOP^
034116  254 04 0 00 034117              HALT    .+1                     ;Test failed if program halts here
034117  324 00 0 00 034120              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that hrlm places c(ac-right) into e-left and does not
                                ;Modify e-right.  in this case, ac is preloaded with 0 and e is preloaded
                                ;With -1,,-1.  then, hrlm is executed.  e is then checked for 0,,-1.  if
                                ;C(e)=0,,-1, this test passes.

034120  400 00 0 00 000000      C17500: SETZ                            ;Preload ac with 0
034121  474 01 0 00 000000              SETO    1,                      ;Preload e with -1,,-1
034122  506 00 0 00 000001              HRLM    1                       ;*Hrlm should place 0,,-1 into e
034123  302 01 0 00 777777              CAIE    1,-1                    ;Pass if c(e)=0,,-1
                                        STOP^
034124  254 04 0 00 034125              HALT    .+1                     ;Test failed if program halts here
034125  324 00 0 00 034126              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
                                        ;This test verifies that hrrs does not modify e.  e is preloaded
                                        ;With 0 and ac is preloaded with -1,,-1.  hrrs is executed; then
                                        ;E is checked.  if c(e) does not change, this test passes.

034126  474 00 0 00 000000      C17600: SETO                            ;Preload ac with -1,,-1
034127  400 01 0 00 000000              SETZ    1,                      ;Preload e with 0,,0
034130  543 00 0 00 000001              HRRS    1                       ;*Hrrs should place 0,,0 into e
034131  332 00 0 00 000001              SKIPE   1
                                        STOP^
034132  254 04 0 00 034133              HALT    .+1                     ;Test failed if program halts here
034133  324 00 0 00 034134              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that hrrzm places c(ac-right) into e-right and places
                                        ;Zeros into e-left.  in this case, ac=e=ac1 and c(ac)=c(e)=-1,,0.  hrrzm
                                        ;Is executed and ac1 is checked for 0.  if ac1=0, this test passes.

034134  474 00 0 00 000000      C17700: SETO                            ;Preload ac0 with -1,,-1
034135  205 01 0 00 777777              MOVSI   1,-1                    ;Preload ac1 with -1,,0
034136  552 01 0 00 000001              HRRZM   1,1                     ;*Hrrzm should place 0 into ac1
034137  332 00 0 00 000001              SKIPE   1                       ;Pass if c(ac1)=0
                                        STOP^
034140  254 04 0 00 034141              HALT    .+1                     ;Test failed if program halts here
034141  324 00 0 00 034142              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                        ;This test verifies that jfcl 17,,+1 never jumps and does not modify c(ac0).
                                        ;First, ac0 is preloaded; then, jfcl is executed.  if ac0 is modified or
                                        ;Jfcl skips, this test fails.

034142  400 00 0 00 000000      C20000: SETZ                            ;Clear ac0
034143  255 17 0 00 034144              JFCL    17,,+1                  ;*Jfcl should not jump or modify c(ac0).
034144  332 00 0 00 000000              SKIPE                           ;Pass if c(ac0)=0 and jfcl did not jump
                                        STOP^
034145  254 04 0 00 034146              HALT    .+1                     ;Test failed if program halts here
034146  324 00 0 00 034147              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that xorm performs the logical exclusive or function
                                        ;Between c(ac) and c(e) and places the result into e.  in this case,
                                        ;Ac and e are preloaded with -1,,-1; then, xorm is executed.  if the
                                        ;Result in e is 0, the test passes.

034147  477 00 0 00 000001      C20100: SETOB   1                       ;Preload ac,e with -1,,-1
034150  432 00 0 00 000001              XORM    1                       ;*Xorm should place 0 into e
034151  302 01 0 00 000000              CAIE    1,0                     ;Pass if c(e)=0
                                        STOP^
034152  254 04 0 00 034153              HALT    .+1                     ;Test failed if program halts here
034153  324 00 0 00 034154              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
5970                                    ;This test verifies that setzb places zeros into both ac and e.
5971                                    ;After setzb is executed, both ac and e are checked for 0.  if
5972                                    ;Either ac or e contains any ones, this test fails.
5973
5974  034154  403 00 0 00 000001        C20200: SETZB   1                       ;*Setzb should place zeroes in both ac and e
5975  034155  316 00 0 00 034513                CAMN    [0]                     ;Fail if c(ac) is non-zero
5976  034156  312 00 0 00 000001                CAME    1                       ;Fail if c(e) is non-zero
5977                                            STOP^
5978  034157  254 04 0 00 034160                HALT    .+1                     ;Test failed if program halts here
5979  034160  324 00 0 00 034161                JUMPA   .+1                     ;To loop change this instruction^
5980
5981                                    ;**********
5982
5983                                    ;This test verifies that setab places c(ac) into both ac and e.
5984                                    ;First, ac is preloaded with -1,,-1 and e is preloaded with 0;
5985                                    ;Then, setab is executed.  both ac and e are checked for -1,,-1
5986                                    ;If either ac or e contain any zeros, this test fails.
5987
5988  034161  400 01 0 00 000000        C20300: SETZ    1,                      ;Preload e with 0
5989  034162  474 00 0 00 000000                SETO                            ;Preload ac with -1,,-1
5990  034163  427 00 0 00 000001                SETAB   1                       ;*Setab should place -1,,-1 into both ac and e
5991  034164  316 00 0 00 034443                CAMN    [-1]                    ;Fail if c(ac) is not -1,-1
5992  034165  312 00 0 00 000001                CAME    1                       ;Fail if c(e) is not -1,,-1
5993                                            STOP^
5994  034166  254 04 0 00 034167                HALT    .+1                     ;Test failed if program halts here
5995  034167  324 00 0 00 034170                JUMPA   .+1                     ;To loop change tnis instruction^
5996
5997                                    ;**********
```

```
                        SUBTTL  XCT INSTRUCTION - BASIC TESTS

                        ;**********

                        ;This test verifies that xct will execute the instruction specified by c(e).
                        ;In this case, c(e) specifies a movei instruction.  after executing movei,
                        ;Control should return to the next sequential instruction following xct. this
                        ;Test passes if control returns to the next sequential instruction following xct.

034170  403 00 0 00 000001      C20400: SETZB   1                       ;Clear ac0 and ac1
034171  256 00 0 00 034517              XCT     [MOVEI 1,.+2]           ;*Xct should return control to next instruction
034172  334 00 0 00 000000              SKIPA
                                        STOP^
034173  254 04 0 00 034174              HALT    .+1                     ;Test failed if program halts here
034174  324 00 0 00 034175              JUMPA   .+1                     ;To loop change this instruction^

                        ;**********

                        ;This test verifies that xct will execute the instruction specified by c(e)
                        ;In this case, c(e) specifies a movei instruction. after executing movei,
                        ;The ac specified by movei is checked for 0,,1 (the expected result).  if
                        ;C(ac)=0,,1, this test passes.

034175  403 00 0 00 000001      C20500: SETZB   1                       ;Clear ac
034176  256 00 0 00 034520              XCT     [MOVEI 1,1]             ;*Xct of movei should place 1 in the ac
034177  302 01 0 00 000001              CAIE    1,1                     ;Pass if c(ac)=1
                                        STOP^
034200  254 04 0 00 034201              HALT    .+1                     ;Test failed if program halts here
034201  324 00 0 00 034202              JUMPA   .+1                     ;To loop change this instruction^

                        ;**********
```

```
                                ;This test verifies that a nest of xct instructions will execute the
                                ;Instruction specified by the most nested xct and return control to the
                                ;Next sequential instruction following the first xct. in this case, the
                                ;Executed instruction is movei.  after executing the movei, c(ac) is
                                ;Checked for 0,,-1 (the expected result).  if c(ac)=0,,-1, this test passes.

034202  403 00 0 00 000001      C20600: SETZB   1                       ;Clear ac
034203  256 00 0 00 034525              XCT     [XCT[XCT[XCT[XCT[MOVEI 1,-1]]]]] ;*Nested xct of movei
                                                                        ;Should place 0,,-1 into ac
034204  302 01 0 00 777777              CAIE    1,-1                    ;Pass if c(ac)=0,,-1
                                        STOP^
034205  254 04 0 00 034206              HALT    .+1                     ;Test failed if program halts here
034206  324 00 0 00 034207              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that xct will not modify an ac which is not specified by
                                ;The executed instruction.  in this case, ac0 is cleared and then checked for
                                ;Zero after the xct instruction is executed.  ac0 should not be modified.

034207  403 00 0 00 000001      C20700: SETZB   1                       ;Clear ac0,ac1
034210  256 00 0 00 034526              XCT     [MOVE 1,[-1]]           ;*Xct should not modify ac0
034211  332 00 0 00 000000              SKIPE                           ;Pass if ac0 was not modified
                                        STOP^
034212  254 04 0 00 034213              HALT    .+1                     ;Test failed if program halts here
034213  324 00 0 00 034214              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that xct of skipa should return control to
                                ;The second sequential instruction following xct.

034214  256 00 0 00 034527      C21000: XCT     [SKIPA]                 ;Xct of skipa should return control to .+2
                                        STOP^
034215  254 04 0 00 034216              HALT    .+1                     ;Test failed if program halts here
034216  324 00 0 00 034217              JUMPA   .+1                     ;To loop change this instruction^
```

```
                                SUBTTL  INDIRECT ADDRESSING - BASIC TESTS

                                ;**********

                                ;This test verifies that indirect addressing works when both e and @e
                                ;Are within the ac range.  the instruction specifying indirect addressing
                                ;Is a move.  after move is executed, c(ac) is checked for 0, the  initial
                                ;Contents of the indirect address.

034217  476 00 0 00 000001      C21100: SETOM   1               ;Preload ac with -1,,-1
034220  201 07 0 00 000003              MOVEI   7,3             ;Setup direct address with indirect address
034221  402 00 0 00 000003              SETZM   3               ;Preload indirect address with 0
034222  200 01 1 00 000007              MOVE    1,@7            ;*Fwt from indirect addresss should
                                                                ;Place 0 into the ac
034223  332 00 0 00 000001              SKIPE   1               ;Pass if c(ac)=0
                                        STOP^
034224  254 04 0 00 034225              HALT    .+1             ;Test failed if program halts here
034225  324 00 0 00 034226              JUMPA   .+1             ;To loop change this instruction^

                                ;**********

                                ;This test verifies that indirect addressing works when both e and @e
                                ;Are within the ac range. the instruction specifying indirect addressing
                                ;Is a move.  after move is executed, c(ac) is checked for -1,,-1, the
                                ;Initial contents of the indirect address.

034226  402 00 0 00 000001      C21200: SETZM   1               ;Preload ac with -1,,-1
034227  201 07 0 00 000003              MOVEI   7,3             ;Setup direct address with indirect address
034230  476 00 0 00 000003              SETOM   3               ;Preload indirect address with -1,,-1
034231  200 01 1 00 000007              MOVE    1,@7            ;*Fwt from indirect address should
                                                                ;Place -1,,-1 into the ac
034232  312 01 0 00 034443              CAME    1,[-1,,-1]      ;Pass if c(ac)=-1,,-1
                                        STOP^
034233  254 04 0 00 034234              HALT    .+1             ;Test failed if program halts here
034234  324 00 0 00 034235              JUMPA   .+1             ;To loop change this instruction^

                                ;**********
```

```
6102                                    ;This test verifies that indirect addressing works when both e and @e
6103                                    ;Are within the ac range. the instruction specifying indirect addressing
6104                                    ;Is a move. after move is executed, c(ac) is checked for 707070,,707070,
6105                                    ;The initial contents of the indirect address.
6106
6107  034235  402 00 0 00 000001        C21300: SETZM   1                       ;Preload ac with 0
6108  034236  201 07 0 00 000003                MOVEI   7,3                     ;Setup direct address with indirect address
6109  034237  200 03 0 00 034530                MOVE    3,[707070,,707070] ;Preload indirect address with 707070,,707070
6110  034240  200 01 1 00 000007                MOVE    1,@7                    ;*Fwt from indirect address should
6111                                                                            ;Place 707070,,707070 into the ac
6112  034241  312 01 0 00 034530                CAME    1,[707070,,707070]      ;Pass if c(ac)=707070,,707070
6113                                            STOP^
6114  034242  254 04 0 00 034243                HALT    .+1                     ;Test failed if program halts here
6115  034243  324 00 0 00 034244                JUMPA   .+1                     ;To loop change this instruction^
6116
6117                                    ;**********
6118
6119                                    ;This test verifies that indirect addressing works when e is within the ac
6120                                    ;Range and @e is beyond the ac range.  the instruction specifying indirect
6121                                    ;Addressing is a move.  after move is executed, c(ac) is checked for
6122                                    ;707070,,707070, The initial contents of the indirect address.
6123
6124  034244  254 00 0 00 034246        C21400: JRST    .+2
6125  034245  707070  707070                    XWD     707070,707070           ;Indirect address and its data
6126  034246  402 00 0 00 000001                SETZM   1                       ;Preload ac with 0
6127  034247  201 07 0 00 034245                MOVEI   7,C21400+1              ;Setup direct address with indirect address
6128  034250  200 01 1 00 000007                MOVE    1,@7                    ;*Fwt from indirect address should
6129                                                                            ;Place 707070,,707070 into ac
6130  034251  312 01 0 00 034245                CAME    1,C21400+1              ;Pass if c(ac)=707070,,707070
6131                                            STOP^
6132  034252  254 04 0 00 034253                HALT    .+1                     ;Test failed if program halts here
6133  034253  324 00 0 00 034254                JUMPA   .+1                     ;To loop change this instruction^
6134
6135                                    ;**********
```

```
                                ;This test verifies that indirect addressing works when both e and @e
                                ;Are beyond the ac range. the instruction specifying indirect addressing
                                ;Is a move. after move is executed, c(ac) is checked for 202020,,202020,
                                ;The initial contents of the indirect address.

034254  254 00 0 00 034257      C21500: JRST    .+3
034255  000000  034256                  .+1                     ;Direct address and its data
034256  202020  202020                XWD     202020,202020     ;Indirect address and its data
034257  402 00 0 00 000001              SETZM   1               ;Preload ac with 0
034260  200 01 1 00 034255              MOVE    1,@C21500+1     ;*Fwt from indirect address should
                                                                ;Place 202020,,202020 into ac
034261  312 01 0 00 034256              CAME    1,C21500+2      ;Pass if c(ac)=202020,,202020
                                        STOP^
034262  254 04 0 00 034263              HALT    .+1             ;Test failed if program halts here
034263  324 00 0 00 034264              JUMPA   .+1             ;To loop change this instruction^

                                ;**********

                                ;This test verifies that indirect addressing works when both e and @e
                                ;Are beyond the ac range.  the instruction specifying indirect
                                ;Addressing is a came.

034264  254 00 0 00 034267      C21600: JRST    .+3
034265  000000  034266                  .+1                     ;Direct address and its data
034266  272727  272727                XWD     272727,272727     ;Indirect address and its data
034267  200 01 0 00 034266              MOVE    1,C21600+2      ;Preload ac
034270  312 01 1 00 034265              CAME    1,@C21600+1     ;*Came of data from indirect address - non-ac range
                                        STOP^
034271  254 04 0 00 034272              HALT    .+1             ;Test failed if program halts here
034272  324 00 0 00 034273              JUMPA   .+1             ;To loop change this instruction^

                                ;**********

                                ;This test verifies that indirect addressing works when e is within the ac
                                ;Range and @e is beyond the ac range.  the instruction specifying indirect
                                ;Addressing is a came.

034273  254 00 0 00 034275      C21700: JRST    .+2
034274  252525  252525                XWD     252525,252525     ;Indirect address and its data
034275  201 07 0 00 034274              MOVEI   7,C21700+1      ;Setup direct address with indirect address
034276  200 01 0 00 034274              MOVE    1,C21700+1      ;Setup ac
034277  312 01 1 00 000007              CAME    1,@7            ;*Came if data from indirect address - ac range
                                        STOP^
034300  254 04 0 00 034301              HALT    .+1             ;Test failed if program halts here
034301  324 00 0 00 034302              JUMPA   .+1             ;To loop change this instruction^
```

```
6181                                    SUBTTL  TEST INDIRECT ADDRESSING WITH INDEXING
6182
6183                                    ;Setup index registers
6184
6185  034302  201 01 0 00 777774                MOVEI   1,-4
6186  034303  201 03 0 00 000002                MOVEI   3,2
6187  034304  201 04 0 00 000010                MOVEI   4,10
6188  034305  201 05 0 00 000001                MOVEI   5,1
6189  034306  201 06 0 00 000005                MOVEI   6,5
6190  034307  201 07 0 00 000007                MOVEI   7,7
6191  034310  201 10 0 00 000004                MOVEI   10,4
6192  034311  201 11 0 00 777772                MOVEI   11,-6
6193  034312  201 12 0 00 000005                MOVEI   12,5
6194  034313  201 13 0 00 000002                MOVEI   13,2
6195
6196  034314  254 00 0 00 034345                JRST    C22000                  ;Resume test
6197
6198                                    ;Indirect addressing/indexing test table
6199
6200                                    ;;;;;;;;;;;         ;;;;;;;;;;         ;;;;;;;;;;
6201
6202                                    ;Do not modify this table or tests c21700 thru c22600 independently !
6203
6204                                    ;;;;;;;;;;         ;;;;;;;;;;         ;;;;;;;;;;
6205
6206  034315  000003  034320        E217:           E217A(3)
6207  034316  000020  034323        E220:           @E220A
6208  034317  220220  220220        E220B:  220220,,220220
6209  034320  000020  034325        E217A:          @E221A                  ;E221-4
6210  034321  221221  221221        E221B:  221221,,221221
6211  034322  217217  217217        E222A:  217217,,217217                  ;E217a+2
6212  034323  000000  034317        E220A:          E220B
6213  034324  000000  034324        E221:           E221
6214  034325  000000  034321        E221A:          E221B
6215  034326  000000  034324        E222:           E221
6216  034327  223223  223223        E223A:  223223,,223223
6217  034330  000004  034330        E224A:          E224A(4)                ;E223-6
6218  034331  222222  222222                        222222,,222222          ;E222a+7
6219  034332  000000  034332        E225:           E225
6220  034333  000007  034322                        E222A(7)                ;E222+5
6221  034334  000020  034337                        @E225A                  ;E225+2
6222  034335  225225  225225        E225B:  225225,,225225
6223  034336  000000  034336        E223:           E223
6224  034337  000000  034335        E225A:          E225B
6225  034340  224224  224224                        224224,,224224          ;E224a+10
6226  034341  226226  226226        E226B:  226226,,226226
6227  034342  000000  034327                        E223A                   ;E223+4
6228  034343  000025  034343        E226A:          @E226A(5)               ;E223+5
6229  034344  000000  034341                        E226B                   ;E226a+1
```

```
                                ;This test verifies that indirect addressing in combination with indexing
                                ;Functions correctly.  in  this  case, move 2,@e217  is  tested  where
                                ;C(e217)=e217a(3) and c(3)=0,,2.  hence, the result in the ac should be
                                ;C(e217a+2)=217217,,217217 .

034345  476 00 0 00 000002      C22000: SETOM   2                       ;Initialize ac
034346  200 02 1 00 034315              MOVE    2,@E217                 ;Test indirect addressing with indexing
034347  312 02 0 00 034322              CAME    2,E217A+2               ;Pass if c(ac)=217217,,217217
                                        STOP^
034350  254 04 0 00 034351              HALT    .+1                     ;Test failed if program halts here
034351  324 00 0 00 034352              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that indirect addressing in combination with indexing
                                ;Functions  correctly.   in  this  case,  move 2,@e220  is tested where
                                ;C(e220)=@e220a and c(e220a)=e220b.  hence, the result in the ac should
                                ;Be c(e220b)=220220,,220220 .

034352  402 00 0 00 000002      C22100: SETZM   2                       ;Initialize ac
034353  200 02 1 00 034316              MOVE    2,@E220                 ;Test indirect addressing with indexing
034354  312 02 0 00 034317              CAME    2,E220B                 ;Pass if c(ac)=220220,,220220
                                        STOP^
034355  254 04 0 00 034356              HALT    .+1                     ;Test failed if program halts here
034356  324 00 0 00 034357              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
                                        ;This test verifies that indirect addressing in combination with indexing
                                        ;Functions correctly.  in this case,e221(1) 2,@e217 is tested where c(1)=-4
                                        ;And e221-4=e217a.  hence, the result in the ac should be
                                        ;C(e217a)=@e221a=20,,e221a .

034357  476 00 0 00 000002      C22200: SETOM   2                       ;Initialize ac
034360  200 02 0 01 034324              MOVE    2,E221(1)               ;Test indirect addressing with indexing
034361  312 02 0 00 034320              CAME    2,E217A                 ;Pass if c(ac)=@e221a=20,,e221a
                                        STOP^
034362  254 04 0 00 034363              HALT    .+1                     ;Test failed if program halts here
034363  324 00 0 00 034364              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********

                                        ;This test verifies that indirect addressing in combination with indexing
                                        ;Functions correctly. in this case, move 2,e222(6) is tested where c(6)=5
                                        ;Hence, the result in the ac should be c(e222+5)=e222a(7)=7,,e222a .

034364  402 00 0 00 000002      C22300: SETZM   2                       ;Initialize ac
034365  200 02 0 06 034326              MOVE    2,E222(6)               ;Test indirect addressing with indexing
034366  312 02 0 00 034333              CAME    2,E222+5                ;Pass if c(ac)=e222a(7)=7,,e222a
                                        STOP^
034367  254 04 0 00 034370              HALT    .+1                     ;Test failed if program halts here
034370  324 00 0 00 034371              JUMPA   .+1                     ;To loop change this instruction^

                                        ;**********
```

```
                                ;This test verifies that indirect addressing in combination with indexing
                                ;Functions correctly. in this case, move 2,@e223(10) is tested where c(10)=4
                                ;And c(e223+4)=e223a.  hence, the result in the ac should be
                                ;C(e223a)=223223,,223223 .

034371  476 00 0 00 000002      C22400: SETOM   2                       ;Initialize ac
034372  200 02 1 10 034336              MOVE    2,@E223(10)             ;Test indirect addressing with indexing
034373  312 02 0 00 034327              CAME    2,E223A                 ;Pass if c(ac)=223223,,223223
                                        STOP^
034374  254 04 0 00 034375              HALT    .+1                     ;Test failed if program halts here
034375  324 00 0 00 034376              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********

                                ;This test verifies that indirect addressing in combination with indexing
                                ;Functions correctly.  in this case, move 2,@e223(11) is tested where
                                ;C(11)=-6, c(e223-6)=e224a(4) and c(4)=10.  hence, the result in the ac
                                ;Should be c(e224a+10)=224224,,224224 .

034376  402 00 0 00 000002      C22500: SETZM   2                       ;Initialize ac
034377  200 02 1 11 034336              MOVE    2,@E223(11)             ;Test indirect addressing with indexing
034400  312 02 0 00 034340              CAME    2,E224A+10              ;Pass if c(ac)=224224,,224224
                                        STOP^
034401  254 04 0 00 034402              HALT    .+1                     ;Test failed if program halts here
034402  324 00 0 00 034403              JUMPA   .+1                     ;To loop change this instruction^

                                ;**********
```

```
6310                                    ;This test verifies that indirect addressing in combination with indexing
6311                                    ;functions correctly.  in this case, move 2,@e225(13) is tested where c(13)=2,
6312                                    ;C(e225+2)=@e225a and c(e225a)=e225b.  hence, the result in the ac should be
6313                                    ;C(e225b)=225225,,225225.
6314
6315  034403  476 00 0 00 000002      C22600: SETOM   2                       ;Initialize ac
6316  034404  200 02 1 13 034332              MOVE    2,@E225(13)             ;Test indirect addressing with indexing
6317  034405  312 02 0 00 034335              CAME    2,E225B                 ;Pass if c(ac)=225225,,225225
6318                                          STOP^
6319  034406  254 04 0 00 034407              HALT    .+1                     ;Test failed if program halts here
6320  034407  324 00 0 00 034410              JUMPA   .+1                     ;To loop change this instruction^
6321
6322                                    ;**********
6323
6324                                    ;This test verifies that indirect addressing in combination with indexing
6325                                    ;functions correctly.  in this case, move 2,@e223(12) is tested where
6326                                    ;C(12)=5, c(e223+5)=@e226a(5), c(5)=1 and c(e226a+1)=e226b.  hence, the
6327                                    ;Result in the ac should be c(e226b)=226226,,226226.
6328
6329  034410  402 00 0 00 000002      C22700: SETZM   2                       ;Initialize ac
6330  034411  200 02 1 12 034336              MOVE    2,@E223(12)             ;Test indirect addressing with indexing
6331  034412  312 02 0 00 034341              CAME    2,E226B                 ;Pass if c(ac)=226226,,226226
6332                                          STOP^
6333  034413  254 04 0 00 034414              HALT    .+1                     ;Test failed if program halts here
6334  034414  324 00 0 00 034415              JUMPA   .+1                     ;To loop change this instruction^
6335
6336                                    ;**********
6337
6338
6339  034415  402 00 0 00 034572      ENDIT:  SETZM   TNUMB#
6340  034416  254 00 0 00 030057              JRST    BEGEND
```

```
                                        SUBTTL  *STOR* RESERVED STORAGE, SEPT 18,1979

                                        ;PROGRAM LITERALS

                                                XLIST
                                        IFNDEF  $LPAPER,<LIST>
                                                LIT
```

6347	034417		
6348	034417	000001	000001
6349	034420	254 00 0	00 030715
6350	034421	000001	000000
6351	034422	000002	000000
6352	034423	000004	000000
6353	034424	000010	000000
6354	034425	000020	000000
6355	034426	000040	000000
6356	034427	000100	000000
6357	034430	000200	000000
6358	034431	000400	000000
6359	034432	001000	000000
6360	034433	002000	000000
6361	034434	004000	000000
6362	034435	010000	000000
6363	034436	020000	000000
6364	034437	040000	000000
6365	034440	100000	000000
6366	034441	200000	000000
6367	034442	400000	000000
6368	034443	777777	777777
6369	034444	777777	777776
6370	034445	777777	777775
6371	034446	777777	777773
6372	034447	777777	777767
6373	034450	777777	777757
6374	034451	777777	777737
6375	034452	777777	777677
6376	034453	777777	777577
6377	034454	777777	777377
6378	034455	777777	776777
6379	034456	777777	775777
6380	034457	777777	773777
6381	034460	777777	767777
6382	034461	777777	757777
6383	034462	777777	737777
6384	034463	777777	677777
6385	034464	777777	577777
6386	034465	777777	377777
6387	034466	777776	777777
6388	034467	777775	777777
6389	034470	777773	777777
6390	034471	777767	777777
6391	034472	777757	777777
6392	034473	777737	777777
6393	034474	777677	777777
6394	034475	777577	777777
6395	034476	777377	777777

```
6396  034477  776777  777777
6397  034500  775777  777777
6398  034501  773777  777777
6399  034502  767777  777777
6400  034503  757777  777777
6401  034504  737777  777777
6402  034505  677777  777777
6403  034506  577777  777777
6404  034507  377777  777777
6405  034510  777777  000000
6406  034511  000000  777777
6407  034512  252525  252525
6408  034513  000000  000000
6409  034514  254 04 0 00 032717
6410  034515  377777  377777
6411  034516  400000  400000
6412  034517  201 01 0 00 034173
6413  034520  201 01 0 00 000001
6414  034521  201 01 0 00 777777
6415  034522  256 00 0 00 034521
6416  034523  256 00 0 00 034522
6417  034524  256 00 0 00 034523
6418  034525  256 00 0 00 034524
6419  034526  200 01 0 00 034443
6420  034527  334 00 0 00 000000
6421  03453C  707070  707070
6422                                    LIST
6423  034531  000000  000000    ENDSLD: 0
6424
6425                            IFDEF   DEBUG,<
6426  034532                    PATCH:  BLOCK DEBUG             ;PATCHING AREA
6427                            >
6428
6429                            ;PROGRAM VARIABLES
6430  034572                            VAR
6431
6432                            IFDEF   PGMEND,<
6433  034573  000000  000000    END:    0
6434                    030000          END     BEGIN   >

NO ERRORS DETECTED

PROGRAM BREAK IS 000000
ABSOLUTE BREAK IS 034574
CPU TIME USED 00:18.819

23P CORE USED
```

C14600	5528#		
C14700	5541#		
C1500	3007#		
C15000	5552#		
C15100	5564#		
C15200	5577#		
C15300	5589#		
C15400	5601#		
C15500	5617#		
C15600	5630#		
C15700	5643#		
C1600	3426#		
C16000	5655#		
C16100	5668#		
C16200	5681#		
C16201	5727#		
C16202	5754#		
C16400	5785#		
C16500	5798#		
C16600	5810#		
C16700	5823#		
C1700	3854#		
C17000	5837#		
C17100	5855#		
C17200	5868#		
C17300	5882#		
C17400	5894#		
C17500	5908#		
C17600	5921#		
C17700	5935#		
C200	901#		
C2000	4055#		
C20000	5948#		
C20100	5962#		
C20200	5974#		
C20300	5988#		
C20400	6007#		
C20500	6021#		
C20600	6035#		
C20700	6049#		
C2100	4254#		
C21000	6061#		
C21100	6074#		
C21200	6091#		
C21300	6107#		
C21400	6124#	6127	6130
C21500	6141#	6145	6147
C21600	6158#	6161	6162
C21700	6173#	6175	6176
C2200	4267#		
C22000	6196	6235#	
C22100	6249#		
C22200	6262#		

C22300	6275#
C22400	6288#
C22500	6302#
C22600	6315#
C22700	6329#
C2400	4280#
C2410	4287#
C2700	4299#
C2710	4306#
C[illegible]	913#
C3000	4317#
C3100	4339#
C3110	4354#
C3200	4364#
C3210	4371#
C3300	4384#
C3400	4398#
C3500	4413#
C3600	4429#
C3700	4443#
C400	926#
C4000	4457#
C4100	4472#
C4200	4486#
C4300	4501#
C4400	4516#
C4500	4529#
C4600	4543#
C4700	4556#
C500	941#
C5000	4571#
C5100	4585#
C5200	4599#
C5300	4614#
C5400	4629#
C5500	4645#
C5600	4663#
C5700	4680#
C5701	4695#
C5702	4709#
C600	1258#
C6000	4727#
C6100	4742#
C6200	4757#
C6300	4776#
C6400	4797#
C6500	4818#
C6600	4835#
C6700	4853#
C700	1583#
C7000	4868#
C7100	4888#
C7200	4909#

```
C7300       4931#
C7400       4950#
C7500       4966#
C7600       4979#
C7700       4996#
CCA          360#
CHAIN        112#
CHNOFF       360#
CHNON        360#
CLK          360#
CLKCLR       360#
CLKDIS       360#
CLKENB       360#
CLKU         360#
CLOCKF       428#
CNTLC        445#
CNTRP        360#
COMMA        554#
CONSW        429#
CPOPJ        449#
CPOPJ1       447#
CRLF         242     243     549#
CRLF2        246     247     551#
CRY0         360#   4760    4890    5018    5022
CRY1         360#   4837    5038    5042
CSHFLG       608#
CSHMEM       609#
CTRP         360#
CYCL60       602#
DCK          360#   5476    5480    5493    5502
DDT          695#
DDTLNK       126#    387
DDTSRT       387#
DEBUG         30#   6425    6426
DECVER         4#     10      24     408
DF22F        513#
DIAGMN       376#
DIAGNO       694#
DIAMON       696#
DING          98#
DOLLAR       568#
DONG11       697#    828     841
DSKUPD       413
DTE          701#    828     841
DTE0         702#
DTE1         703#
DTE2         704#
DTE3         705#
E217        6206#   6236
E217A       6206    6209#   6237    6264
E220        6207#   6250
E220A       6207    6212#
E220B       6208#   6212    6251
```

```
E221        6213#   6213    6215    6263
E221A       6209    6214#
E221B       6210#   6214
E222        6215#   6276    6277
E222A       6211#   6220
E223        6223#   6223    6289    6303    6330
E223A       6216#   6227    6290
E224A       6217#   6217    6304
E225        6219#   6219    6316
E225A       6221    6224#
E225B       6222#   6224    6317
E226A       6228#   6228
E226B       6226#   6229    6331
END         6433#
ENDFIX       688#
ENDIT       6339#
ENDSLD      6423#
ERDIAG        37#
ERMORE       462#
ERRPC        433#
ERRTLS       434#
ERSTOP       101#
EXCASB        31#   411
EXCMEM       360#
EXCPFW       360#
EXIOT        360#
FOV          360#   4855    4870    4911    5057    5061    5419
FOVU         360#
FRDLNK       124#
FSELNK       123#
FXU          360#   5437    5441    5454    5463
HYPEN        559#
IADBRK       360#
IADEXC       360#
IADSTP       360#
IADUSR       360#
IAPRC1       360#
IAPRE1       360#
IASRTC       360#
IASRTE       360#
IASRTS       360#
ICNSLL       360#
ICNSLR       360#
IDATAF       360#
IEVNPR       360#
IFMMAN       360#
IINSTF       360#
IIOPFC       360#
IIOPFL       360#
IMAINT       360#
IMGINM       360#
IMGNLO       360#
IMGNOF       360#
```

```
IMGNON      360#
IMIPGD      360#
IMLAPD      360#
INHCSH      109#
INHPAG      106#
INXCLR      360#
INXM        360#
INXSTP      360#
IOCLR       360#
IP50HZ      360#
IPAREN      360#
IPARER      360#
IPRSTP      360#
IPWRFL      360#
IPWRLO      360#
IRQCLR      360#
ISPDOF      360#
ISPDON      360#
ITERAT       65#    406
ITMDIS      360#
ITMENB      360#
ITMOEN      360#
ITMOUT      360#
ITMSET      360#
ITRCH1      466#
ITRCNT      406#    782     803     807     815     8:3     820
IWRITE      360#
JOB41       360#
JOBAPR      36C#
JOBCNI      360#
JOBDDT      360#
JOBFF       360#
JOBOPC      360#
JOBREL      360#
JOBREN      360#
JOBSA       360#
JOBSYM      360#
JOBTPC      360#
JOBUSY      360#
JOBUUO      360#
JOBVER      360#
KA10        360
KAHZ50      114#
KAIFLG      423#
KI10         33#    360
KL10         34#    360
KL10PO      360
KL10PO       35#
KLFLG       424#    783     794     813
KLOLD       322
KNTRP       360#
KTRP        360#
LAPRAL      360#
```

LAPRP1	360#		
LAPRP2	360#		
LAPRP3	360#		
LAPRP4	360#		
LAPRP5	360#		
LAPRP6	360#		
LAPRP7	360#		
LAROVT	360#		
LCASDE	360#		
LCASLD	360#		
LCASLO	360#		
LCASWB	360#		
LCASWD	360#		
LCCASD	360#		
LCHNOF	360#		
LCHNON	360#		
LCIOPF	360#		
LCNTRP	360#		
LCNXER	360#		
LCPAER	360#		
LCPWRF	360#		
LCSBER	360#		
LCSLOA	360#		
LCSLOC	360#		
LCTRP	360#		
LCWSX	360#		
LDATAF	360#		
LDCASD	360#		
LDEXCB	360#		
LDIOPF	360#		
LDLNK	125#	376	378
LDNXER	360#		
LDPAER	360#		
LDPWRF	360#		
LDSBER	360#		
LDUSRB	360#		
LECASD	360#		
LEIOPF	360#		
LENXER	360#		
LEPAER	360#		
LEPWRF	360#		
LESBER	360#		
LEUPFW	360#		
LEVNCD	360#		
LEVNPA	360#		
LEVNPD	360#		
LEXCMP	360#		
LFLGCL	360#		
LFLGDS	360#		
LFLGEN	360#		
LFLGST	360#		
LFP	563#		
LINSTF	360#		

```
LINT        360#
LIOCLR      360#
LIOPFE      360#
LIP         360#
LKNTRP      360#
LKTRP       360#
LLACBL      360#
LLDUSB      360#
LLPRCN      360#
LMUUO       360#
LMUUOP      360#
LNXMEN      360#
LNXMER      360#
LOOPER      100#
LPAREN      360#
LPARER      360#
LPDOVT      360#
LPFWPC      360#
LPGFTR      360#
LPICH1      360#
LPICH2      360#
LPICH3      360#
LPICH4      360#
LPICH5      360#
LPICH6      360#
LPICH7      360#
LPICHA      360#
LPICLR      360#
LPIIP1      360#
LPIIP2      360#
LPIIP3      360#
LPIIP4      360#
LPIIP5      360#
LPIIP6      360#
LPIIP7      360#
LPIOFF      360#
LPION       360#
LPNTRP      360#
LPRCH1      360#
LPRCH2      360#
LPRCH3      360#
LPRCH4      360#
LPRCH5      360#
LPRCH6      360#
LPRCH7      360#
LPTRP       360#
LPWRFE      360#
LPWRFL      360#
LREQSE      360#
LRQCLR      360#
LSBSEN      360#
LSBUSE      360#
LSCASD      360#
```

LSECMO	360#			
LSIOPF	360#			
LSMODE	360#			
LSNTRP	360#			
LSNXER	360#			
LSPAER	360#			
LSPWRF	360#			
LSSBER	360#			
LSTRP	360#			
LTRP3T	360#			
LTRPAE	360#			
LTRPEN	360#			
LUSCMP	360#			
LUUO	360#			
LUUO1	493	494		
LUUO10	493	498		
LUUO11	493	498		
LUUO12	493	499		
LUUO13	493	499		
LUUO14	493	500		
LUUO15	493	500		
LUUO16	493	501		
LUUO17	493	501		
LUUO2	493	495		
LUUO20	493	502		
LUUO21	493	502		
LUUO22	493	503		
LUUO23	493	503		
LUUO24	493	504		
LUUO25	493	504		
LUUO26	493	505		
LUUO27	493	505		
LUUO3	493	495		
LUUO30	493	506		
LUUO31	493	506		
LUUO32	493	507		
LUUO33	493	507		
LUUO4	493	496		
LUUO5	493	496		
LUUO6	493	497		
LUUO7	493	497		
LUUOI	360#			
LWRITE	360#			
MAPNEW	514#			
MARGIN	436#			
MCNVER	3#	10	24	408
MEMLOW	516#			
MEMMAP	38#	414		
MEMSIZ	517#			
MEMTOT	515#			
MINUS	558#			
MODDVC	108#			
MODDVL	307	307#	409	

MODDVU	308	308#	410			
MODLNK	127#	373				
MONCTL	426#	823	850			
MONFLG	425#	776				
MONTEN	427#	781	809	835		
MONTYP	640#					
MPVU	360#					
MUUO	360#					
MUUOPC	360#					
NOPNT	96#					
NXMU	360#					
OPRSEL	110#					
OPTIME	528#					
P	153	154	155	156	360#	448
PAG	360#					
PALERS	102#					
PARCLR	360#					
PARDIS	360#					
PAREA0	58#					
PAREA1	59#	404				
PAREA2	60#	405				
PAREA3	61#	402				
PAREA4	62#	403				
PAREA5	63#	415				
PAREA6	64#	416				
PAREA7	415#					
PAREA8	416#					
PARENB	360#					
PARU	360#					
PASCNT	430#	802				
PATCH	6426#					
PDISF	525#					
PDLOVU	360#					
PDOVTP	360#					
PERIOD	555#					
PFSTRT	382#					
PGFTRP	360#					
PGMEND	36#	6432				
PGMNAM	407	851	854#			
PI	815					
PICHN1	360#					
PICHN2	360#					
PICHN3	360#					
PICHN4	360#					
PICHN5	360#					
PICHN6	360#					
PICHN7	360#					
PICHNA	360#					
PICLR	360#					
PIOFF	360#					
PION	360#					
PLIST	594#	594				
PLISTE	594	596#				

```
PLISTS      595#
PLUS        560#
PNTENB      524#
PNTEXT      403#
PNTFLG      523#
PNTINH      526#
PNTLPT       97#
PNTNAM      402#
PNTRP       360#
PNTSPC      527#
PTRP        360#
PVPAGI      621#
PWFCLR      360#
QUEST       566#
RADIX       569#
RADLSC      571#
RADLSP      570#
RANDBS      404#
REENTR      384#
RELIAB      104#
REPT        360#
REPT1       360#
REPTU       482#
REQSET      360#
RESRT1      459#
RESRT2      460#
RETURN      391#    811     818     836
RSTART       93#
RTP         564#
RUNFLG      431#
SADR1        43#    380
SADR10       52#    396
SADR11       53#    397
SADR2        44#    382
SADR3        45#    384
SADR4        46#
SADR5        47#    445
SADR6        48#    446
SADR7        49#    393
SADR8        50#    394
SADR9        51#    395
SBINIT      166     390#
SCOPE       483#
SENSE1      360#
SENSE2      360#
SENSE3      360#
SENSE4      360#
SENSE5      360#
SENSE6      360#
SFSTRT      380#
SLASH       567#
SM10        671#
SMLUSR      360#
```

SN
933# 999 999# 1015 1015# 1031 1031# 1047 1047# 1063 1063# 1079 1079# 1095
1095# 1111 1111# 1127 1127# 1143 1143# 1159 1159# 1175 1175# 1191 1191# 1207
1207# 1223 1223# 1239 1239# 1249# 1332 1332# 1349 1349# 1366 1366# 1383 1383#
1400 1400# 1417 1417# 1434 1434# 1451 1451# 1468 1468# 1485 1485# 1502 1502#
1519 1519# 1536 1536# 1553 1553# 1570 1570# 1580# 1603 1603# 1619 1619# 1635
1635# 1651 1651# 1667 1667# 1683 1683# 1699 1699# 1715 1715# 1731 1731# 1747
1747# 1763 1763# 1779 1779# 1795 1795# 1811 1811# 1827 1827# 1837# 1863 1863#
1881 1881# 1899 1899# 1917 1917# 1935 1935# 1953 1953# 1971 1971# 1989 1989#
2007 2007# 2025 2025# 2043 2043# 2061 2061# 2079 2079# 2097 2097# 2115 2115#
2148# 2179 2179# 2201 2201# 2223 2223# 2245 2245# 2267 2267# 2289 2289# 2311
2311# 2333 2333# 2355 2355# 2377 2377# 2399 2399# 2421 2421# 2443 2443# 2465
2465# 2487 2487# 2509 2509# 2531 2531# 2553 2553# 2567# 2598 2598# 2620 2620#
2642 2642# 2664 2664# 2686 2686# 2708 2708# 2730 2730# 2752 2752# 2774 2774#
2796 2796# 2818 2818# 2840 2840# 2862 2862# 2884 2884# 2906 2906# 2928 2928#
2950 2950# 2972 2972# 3004# 3035 3035# 305? 3057# 3079 3079# 3101 3101# 3123
3123# 3145 3145# 3167 3167# 3189 3189# 321 3211# 3233 3233# 3255 3255# 3277
3277# 3299 3299# 3321 3321# 3343 3343# 3365 3365# 3387 3387# 3409 3409# 3423#
3454 3454# 3476 3476# 3498 3498# 3520 3520# 3542 3542# 3564 3564# 3586 3586#
3608 3608# 3630 3630# 3652 3652# 3674 3674# 3696 3696# 3718 3718# 3740 3740#
3762 3762# 3784 3784# 3806 3806# 3828 3828# 3844# 3865 3865# 3877 3877# 3889
3889# 3901 3901# 3913 3913# 3925 3925# 3937 3937# 3949 3949# 3961 3961# 3973
3973# 3985 3985# 3997 3997# 4009 4009# 4021 4021# 4033 4033# 4046# 4066 4066#
4078 4078# 4090 4090# 4102 4102# 4114 4114# 4126 4126# 4138 4138# 4150 4150#
4162 4162# 4174 4174# 4186 4186# 4198 4198# 4210 4210# 4222 4222# 4234 4234#

SNTRP 360#
SPACE 556#
SRTDDT 386#
START 374 779 787 795 799 847#
START1 393#
START2 394#
START3 395#
START4 396#
START5 397#
STARTA 389 797 808 817 824 831 848 852 873#
STRP 360#
SUBLNK 128# 390
SWPTAB 677#
SWTEXR 405#
SYSEXR 378#
TAB 557#
TESTPC 432#
TICKS 435#
TN0 360#
TN1 360#
TNUMB 6339# 6339
TOTALS 94#
TRP3TP 360#
TRPENB 360#
TTNBRF 615#
TTYFIL 531#
TTYSPD 532#
TXTINH 105#
UOLIP 360#

	2644	2646	2649	2665	2665#	2666	2668	2671	2687	2687#	2688	2690	2693	2709
	2709#	2710	2712	2715	2731	2731#	2732	2734	2737	2753	2753#	2754	2756	2759
	2775	2775#	2776	2778	2781	2797	2797#	2798	2800	2803	2819	2819#	2820	2822
	2825	2841	2841#	2842	2844	2847	2863	2863#	2864	2866	2869	2885	2885#	2886
	2888	2891	2907	2907#	2908	2910	2913	2929	2929#	2930	2932	2935	2951	2951#
	2952	2954	2957	2973	2973#	2974	2976	2979	3005#	3036	3036#	3037	3037#	3039
	3042	3058	3058#	3059	3061	3064	3080	3080#	3081	3083	3086	3102	3102#	3103
	3105	3108	3124	3124#	3125	3127	3130	3146	3146#	3147	3149	3152	3168	3168#
	3169	3171	3174	3190	3190#	3191	3193	3196	3212	3212#	3213	3215	3218	3234
	3234#	3235	3237	3240	3256	3256#	3257	3259	3262	3278	3278#	3279	3281	3284
	3300	3300#	3301	3303	3306	3322	3322#	3323	3325	3328	3344	3344#	3345	3347
	3350	3366	3366#	3367	3369	3372	3388	3388#	3389	3391	3394	3410	3410#	3411
	3413	3416	3424#	3455	3455#	3456	3456#	3458	3461	3477	3477#	3478	3480	3483
	3499	3499#	3500	3502	3505	3521	3521#	3522	3524	3527	3543	3543#	3544	3546
	3549	3565	3565#	3566	3568	3571	3587	3587#	3588	3590	3593	3609	3609#	3610
	3612	3615	3631	3631#	3632	3634	3637	3653	3653#	3654	3656	3659	3675	3675#
	3676	3678	3681	3697	3697#	3698	3700	3703	3719	3719#	3720	3722	3725	3741
	3741#	3742	3744	3747	3763	3763#	3764	3766	3769	3785	3785#	3786	3788	3791
	3807	3807#	3808	3810	3813	3829	3829#	3830	3832	3835	3845#	3866	3866#	3868
	3869	3870	3878	3878#	3880	3881	3882	3890	3890#	3892	3893	3894	3902	3902#
	3904	3905	3906	3914	3914#	3916	3917	3918	3926	3926#	3928	3929	3930	3938
	3938#	3940	3941	3942	3950	3950#	3952	3953	3954	3962	3962#	3964	3965	3966
	3974	3974#	3976	3977	3978	3986	3986#	3988	3989	3990	3998	3998#	4000	4001
	4002	4010	4010#	4012	4013	4014	4022	4022#	4024	4025	4026	4034	4034#	4036
	4037	4038	4047#	4067	4067#	4069	4070	4071	4079	4079#	4081	4082	4083	4091
	4091#	4093	4094	4095	4103	4103#	4105	4106	4107	4115	4115#	4117	4118	4119
	4127	4127#	4129	4130	4131	4139	4139#	4141	4142	4143	4151	4151#	4153	4154
	4155	4163	4163#	4165	4166	4167	4175	4175#	4177	4178	4179	4187	4187#	4189
	4190	4191	4199	4199#	4201	4202	4203	4211	4211#	4213	4214	4215	4223	4223#
	4225	4226	4227	4235	4235#	4237	4238	4239						
$$420	746#													
$$421	747#													
$$422	748#													
$$423	749#													
$$424	750#													
$$425	751#													
$$426	752#													
$$427	753#													
$$430	754#													
$$431	755#													
$$432	756#													
$$433	757#													
$$434	758#													
$$435	759#													
$$436	760#													
$$437	761#													
$$500	764#													
$$501	765#													
$$502	766#													
$$503	767#													
$$BEGI	732#													
$$DTE0	792	828#												
$$DTE2	793	841#												

$$LOC	729#	768				
$$MUUO	646#					
$$OUTE	652#					
$$STAR	732	733	772#			
$$TAX1	664#					
$$TAX2	665#					
$$TOGG	658#					
$$UUO	635#					
$ACCO	473#					
$BEGEN	741#					
$BEND1	443#					
$BEND2	444#					
$CHRIN	534#					
$CRLF	536#					
$DDT	710#					
$DEVCH	360#					
$DSKUP	413#					
$DTCHR	722#					
$DTCI	716#					
$DTCLK	715#	790				
$DTCMD	719#	826	839			
$DTF11	718#					
$DTFLG	714#	789	827	829	840	842
$DTMTD	723#					
$DTMTI	724#					
$DTOPR	721#	791				
$DTSEQ	720#					
$DTSWR	726#					
$DTT11	717#					
$DVCH1	360#					
$DVOFF	530#					
$EMODE	411#					
$FFF	538#					
$IBUF	586#					
$INEXT	588#					
$INNM	587#					
$ITRHL	454#					
$ITRX1	455#					
$LPAPE	6346					
$MMAP	414#					
$MODVL	409#					
$MODVU	410#					
$OBUF	577#					
$ONETM	437#					
$OUTEX	579#					
$OUTNM	578#					
$PAPER	360					
$PARER	461#					
$PNAME	407#					
$PSHER	465#					
$PVER	408#					
$RSRTX	457#					
$RSRTY	458#					

$SPAG1	477#	
$SPB1	798	804#
$SPBEN	741	802#
$SPBEX	805	813#
$SPBKL	814	820#
$SPBUS	807#	
$SPEC	738	797#
$SPKLD	821	835#
$START	374#	
$STD	709#	736#
$STKIL	781#	
$STKL	789#	
$STL	711#	
$STM	712#	738#
$SVAPR	475#	
$SVPAG	476#	
$SVPI	474#	
$SVUPC	480#	
$SVUUO	479#	
$TABF	537#	
$TTCHR	533#	
$TWCNT	529#	
$TYPNB	535#	
$UMODE	412#	
$UORTX	452#	
$USRHL	456#	
$UUOER	453#	494
$VTF	539#	
%ACTFL	486#	
%ACTUL	487#	
%COREC	485#	
%CORFL	484#	
%DISCR	488#	
%ERHI1	627#	
%ERHI2	628#	
%ERHI3	629#	
.JB41	360#	
.JBAPR	360#	
.JBCNI	360#	
.JBDDT	360#	
.JBFF	360#	
.JBOPC	360#	
.JBREL	360#	
.JBREN	360#	
.JBSA	360#	
.JBSYM	360#	
.JBTPC	360#	
.JBUSY	360#	
.JBUUO	360#	
.JBVER	360#	

```
MEMSEG      294#
MEMZRO      293#
MODPCP      305#
MODPCU      304#
MTROP       273#
NAME          7#      10
PBELL       248#
PCRL        242#
PCRL2       246#
PCRL2F      247#
PCRLF       243#
PFORCE      250#
PGMINT      166#
PJRST       157#
PMSG        252#
PMSGF       255#
PNT1        208#
PNT11       222#
PNT11F      223#
PNT1F       209#
PNT2        210#
PNT2F       211#
PNT3        212#
PNT3F       213#
PNT4        214#
PNT4F       215#
PNT5        216#
PNT5F       217#
PNT6        218#
PNT6F       219#
PNT7        220#
PNT7F       221#
PNTA        194#
PNTADF      225#
PNTADR      224#
PNTAF       195#
PNTAL       196#
PNTALF      197#
PNTCHF      207#
PNTCHR      206#
PNTCI       204#
PNTCIF      205#
PNTCW       240#
PNTCWF      241#
PNTDCF      233#
PNTDEC      232#
PNTDS       234#
PNTDSF      235#
PNTHW       228#
PNTHWF      229#
PNTMGN      319#
PNTMSF      201#
PNTMSG      200#
```

TTSIXB 189#
TTYINP 190#

SEQ 0182